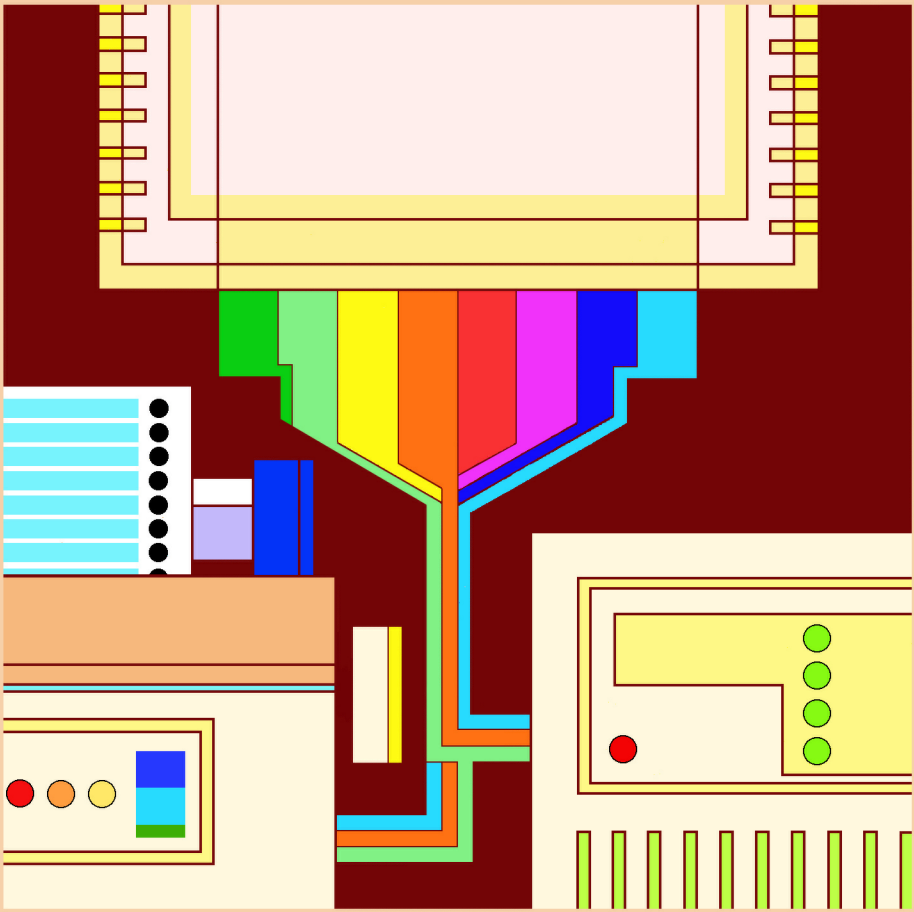
Apple II

Super Serial Card

Installation and Operating Manual

Part 2



Downloaded from www.Apple2Online.com

APPENDIX A
FIRMWARE

This appendix contains the following information:

- an explanation of the Pascal 1.1 firmware card protocol
- a firmware memory map
- a description of the SSC's use of its peripheral slot scratchpad RAM addresses
- a description of the ACIA registers and switch detection registers in the SSC's peripheral I/O space
- a list of firmware entry points & register values
- the actual SSC firmware listings

PASCAL 1.1 FIRMWARE PROTOCOL

The old Apple II Serial Interface Card (SIC) ran under Pascal 1.Ø with three direct firmware entry points, one for each of the three I/O functions it supported:

Address	Contains
$C8ØØ	initialization routine entry point
$C84D	read routine entry point
$C9AA	write routine entry point

New peripheral cards can be "accepted" into the Pascal 1.Ø system by appearing to be a SIC; that is, with these same three entry points and with $38 at $CsØ5 and $18 at $CsØ7 (see Device ID section below).

Pascal 1.1, on the other hand, has a more flexible setup, and also supports more I/O functions. It can make indirect calls to the firmware in a (new) peripheral card through addresses in a branch table in the card's firmware. It also has facilities for uniquely identifying new peripheral I/O devices.

I/O ROUTINE ENTRY POINTS

The I/O routine entry point branch table is located near the beginning of the Cs00 address space (s being the slot number where the peripheral card is installed). This space was chosen instead of the $C800 space, since under BASIC protocol the $Cs00 space is required, while the $C800 space is optional.

The branch table locations that Pascal 1.1 uses are:

Address	Contains
$Cs0D	initialization routine offset (required)
$Cs0E	read routine offset (required)
#Cs0F	write routine offset (required)
$Cs10	status routine offset (required)
$Cs11	$00 if optional offsets follow; non-zero if not
$Cs12	control routine offset (optional)
$Cs13	interrupt handling routine offset (optional)

Notice that $Cs11 contains $00 only if the control and interrupt handling routines are supported by the firmware. (For example, the SSC does not support these two routines, and so location $Cs11 contains a (non-zero) firmware instruction.) Apple II Pascal 1.0 and 1.1 do not support control and interrupt requests, but such requests may be implemented in future versions of the Pascal BIOS and other future Apple II operating systems.

Here are the entry point addresses, and the contents of the 6502 registers on entry to and on exit from Pascal 1.1 I/O routines:

Addr.	Offset for	X Register	Y Register	A Register
$Cs0D	Initialization			
	On entry	$Cs	$s0	
	On exit	error code	(unchanged)	(unchanged)
$Cs0E	Read			
	On entry	$Cs	$s0	
	On exit	error code	(unchanged)	character read
$Cs0F	Write			
	On entry	$Cs	$s0	char. to write
	On exit	error code	(unchanged)	(unchanged)
$Cs10	Status			
	On entry	$Cs	$s0	request (0 or 1)
	On exit	error code	(changed)	(unchanged)

Notes: Request code 0 means, "Are you ready to accept output?"
Request code 1 means, "Do you have input ready?"
On exit, the reply to the status request is in the carry bit: carry clear means "No"; carry set means "Yes."

Table A-1. I/O Routine Offsets and Registers under Pascal 1.1

DEVICE IDENTIFICATION

Pascal 1.1 uses four firmware bytes to identify the peripheral card. Both the identifying bytes and the branch table are near the beginning of the $Cs00 ROM space. The identifiers are listed in Table A-2.

Address	Value
$Cs05	$38 (like the old Serial Interface Card)
$Cs07	$18 (like the old Serial Interface Card)
$Cs0B	$01 (the Generic Signature of new FW cards)
$Cs0C	$ci (the Device Signature; see below)

Table A-2. Bytes Used for Device Identification

The first digit, c, of the Device Signature byte identifies the device class as listed in Table A-3.

Digit	Class
$0	reserved
$1	printer
$2	joystick or other X-Y input device
$3	serial or parallel I/O card
$4	modem
$5	sound or speech device
$6	clock
$7	mass storage device
$8	80-column card
$9	network or bus interface
$A	special purpose (none of the above)
$B-F	reserved for future expansion

Table A-3. Device Class Digit

The second digit, i, of the Device Signature byte is a unique identifier for the card, assigned by Apple Technical Support. For example, the SSC has a Device Signature of $31: the 3 signifies that it is a serial or parallel I/O card, and the 1 is the low-order digit supplied by Apple Technical Support.

Although version 1.1 of Pascal ignores the Device Signature, applications programs can use them to identify specific devices.

SSC FIRMWARE MEMORY USAGE

Table A-4 is an overall map of the locations that the SSC uses, both in the Apple II and in the SSC's own firmware address space.

Addresses	Name of area	Contents
$ØØØØ-$ØØFF	Page Zero	Monitor pointers, I/O hooks, and temporary storage (Table A-5)
$Ø4xx-$Ø7xx (selected locations)	Peripheral Slot Scratchpad RAM	Locations (8 per slot) in Apple's pages $Ø4 through $Ø7. SSC uses all eight of them (Table A-6)
$C0(8+s)Ø - $C0(8+s)F	Peripheral Card I/O Space	Locations (16 per slot) for general I/O; SSC uses 6 bytes (Table A-7)
$Cs00-$CsFF	Peripheral Card ROM Space	One 256-byte page reserved for card in slot s; first page of SSC FW
$C8ØØ-$CFFF	Expansion ROM	Eight 256-byte pages reserved for a 2K ROM or PROM; SSC maps its FW onto $C8ØØ-$CEFF (Table 4-4)

Table A-4. Memory Usage Map

ZERO PAGE LOCATIONS

The SSC makes use of these zero-page locations (Table A-5):

	Address	Name	Description
*	$24	CH	Monitor pointer to current position of cursor on screen
	$26	SLOT16	Usually (slot# x 16); that is, $sØ
	$27	CHARACTER	Input or output character
*	$28	BASL	Monitor pointer to current screen line
	$2A	ZPTMP1	Temporary storage (various uses)
	$2B	ZPTMP2	Temporary storage (various uses)
	$35	ZPTEMP	Temporary storage (various uses)
*	$36	CSWL	BASIC output hook (not for Pascal)
*	$37	CSWH	(high byte of CSW)
*	$38	KSWL	BASIC input hook (not for Pascal)
*	$39	KSWH	(high byte of KSW)
*	$4E	RNDL	random number location, updated when looking for a keypress (not used when initialized by Pascal)

* Not used when Pascal initializes SSC.

Table A-5. Zero-Page Locations Used by SSC

SCRATCHPAD RAM LOCATIONS

The SSC uses the Scratchpad RAM locations as listed in Table A-6.

Address	Field name	Bit(s)	Interpretation
$Ø478+s	DELAYFLG	Ø - 1	<FF> delay selection
		2 - 3	<LF> delay selection
		4 - 5	<CR> delay selection
		6 - 7	Translate option
$Ø4F8+s	HANDSHKE	Ø - 7	Buffer count for handshake (P8A Mode)
	PARAMETER	Ø - 7	Accumulator for FW's command processor
$Ø578+s	STATEFLG	Ø - 2	Command mode when not Ø (Printer and Communications Modes only)
		Ø - 4	Enquire character (P8A Mode); dflt ETX
		3 - 5	Slot to chain to (Communications Mode)
		6	Set to 1 after lowercase input character
		7	Terminal Mode when 1 (Comm Mode)
		7	Enable <CR> gen. when 1 (other 3 modes)
$Ø5F8+s	CMDBYTE	Ø - 6	Printer Mode default is <CTRL-I>; Comm Mode default is <CTRL-A>
		7	Set to 1 to Zap control commands
$Ø678+s	STSBYTE		Status and IORESULT byte (Appendix F)
$Ø6F8+s	CHNBYTE	Ø - 2	Current Apple screen slot (Comm Mode); when slot = Ø, chaining is enabled
		3 - 7	$CsØØ space entry point (Comm Mode)
	PWDBYTE	Ø - 7	Current printer width (other modes); for listing compensation, auto-<CR>
$Ø778+s	BUFBYTE	Ø - 6	One-byte input buffer (Comm Mode); used in conjunction with XOFF recognition
		7	Set to 1 when buffer full (Comm Mode)
	COLBYTE	Ø - 7	Current-column counter for tabbing, etc. (other 3 modes)
$Ø7F8+s	MISCFLG	Ø	Generate <LF> after <CR> when 1
		1	Printer Mode when Ø; Comm Mode when 1
		2	Keyboard input enabled when 1
		3	<CTRL-S> (XOFF), <CTRL-R> and <CTRL-T> input checking when 1
		4	Pascal Op Sys when 1; BASIC when Ø
		5	Discard <LF> input when 1
		6	Enable lowercase and special character generation when 1 (Comm Mode)
		6	Tabbing option on when 1 (Printer Mode)
		7	Echo output to Apple screen when 1

Table A-6. Scratchpad RAM Locations Used by SSC

PERIPHERAL CARD I/O SPACE

There are 16 bytes of I/O space allocated to each slot in the Apple II. Each set begins at address $C0$8$0 + (slot x 16); for example, if the SSC is in slot 3, its group of bytes extends from $C0$B$0 to $C0BF. Table A-7 interprets the 6 bytes the SSC uses.

Address	Register	Bit(s)	Interpretation
$C0$8$1+s$0	DIPSW1	0	SW1-6 is OFF when 1, ON when 0
	(SW1-x)	1	SW1-5 is OFF when 1, ON when 0
		4 - 7	same as above for SW1-4 through SW1-1
$C0$8$2+s$0	DIPSW2	0	Clear To Send (CTS) is true (-) when 0
	(SW2-x)	1 - 3	same as above for SW2-5 through SW2-3
		5 & 7	same as above for SW2-2 & SW2-1
$C0$8$8+s$0	TDREG	0 - 7	ACIA Transmit Register (write)
	RDREG	0 - 7	ACIA Receive Register (read)
$C0$8$9+s$0	STATUS		ACIA Status/Reset Register
		0	Parity error detected when 1
		1	Framing error detected when 1
		2	Overrun detected when 1
		3	ACIA Receive Register full when 1
		4	ACIA Transmit Register empty when 1
		5	Data Carrier Detect (DCD) true when 0
		6	Data Set Ready (DSR) true when 0
		7	Interrupt (IRQ) has occurred when 1
$C0$8A+s$0	COMMAND		ACIA Command Register (read/write)
		0	Data Terminal Ready (DTR): enable (1) or disable (0) receiver and all interrupts
		1	When 1, allow STATUS bit 3 to cause IRQ
		2 - 3	Control transmit interrupt, Request To Send (RTS) level, and transmitter
		4	When 0, normal mode for receiver; when 1, echo mode (but bits 2 and 3 must be 0)
		5 - 7	Control parity (values: Table 2-7)
$C0$8B+s$0	CONTROL		ACIA Control Register (read/write)
		0 - 3	Baud rate: $0 = 16 times external clock; $1 - $F = decimal in Table 2-5
		4	When 1, use baud rate generator; when 0, use external clock (not supported)
		5 - 6	Number of data bits: 8 (bit 5 and 6 = 0) 7 (5 = 1, 6 = 0), 6 (5 = 0, 6 = 1) or 5 (bit 5 and 6 both = 1)
		7	Number of stop bits: 1 (bit 7 = 0); if bit 7 = 1, then 1-1/2 (with 5 data bits, no parity), 1 (8 data plus parity) or 2

Table A-7. SSC Registers in Peripheral Card I/O Space

SSC ENTRY POINTS

This section contains the SSC firmware entry points for the Apple II Monitor, BASIC, Pascal 1.Ø and Pascal 1.1. The Pascal 1.1 entry point offsets conform to the Firmware card protocol outlined in the first section of this appendix.

MONITOR ROM ENTRY POINTS

The SSC uses these entry points in the Monitor ROM, unless Pascal initializes the SSC.

Address	Name	Description
$FDED	COUT	sends a character to output hook (chaining) used for chaining
$FE89	SETKBD	sets KSW to point to keyboard (reset)
$FE93	SETSCR	sets CSW to point to Apple screen (reset)
$FF58	IORTS	known position of an RTS instruction
$FDF6	VIDOUT	sends a character to the Apple screen

Table A-8. Monitor ROM Entry Points Used by SSC

BASIC ENTRY POINTS

Here are the entry point addresses, and the contents of the 65Ø2 registers on entry to and on exit from BASIC I/O routines:

Addr.	Routine	X Register	Y Register	A Register
$CsØØ	Initialization			
	On entry	anything	anything	anything
	On exit	(unchanged)	(unchanged)	character
Notes:	CSW and/or KSW points to $CsØØ. The character in the A register is output unless KSW points to $CsØØ and CSW does not point to $CsØØ.			
$CsØ5	Input			
	On entry	anything	anything	anything
	On exit	(unchanged)	(unchanged)	character in
Notes:	Character in is from ACIA or keyboard.			
$CsØ7	Output			
	On entry	anything	anything	character out
	On exit	(unchanged)	(unchanged)	(changed)
Notes:	Character out is transmitted through the ACIA.			

Table A-9. BASIC Entry Points Used by SSC

PASCAL 1.0 ENTRY POINTS

There are three Pascal 1.Ø entry points: one for initialization, one for read operations, and one for write operations. These entry points are direct addresses.

Addr.	Routine	X Register	Y Register	A Register
$C8ØØ	Initialization			
	On entry	$Cs	$sØ	anything
	On exit	$Cs	$sØ	(unchanged)
Notes:	$C8ØØ space is enabled. Firmware initializes SSC to default values plus SW1 and SW2 selections.			
$C84D	Read			
	On entry	$Cs	$sØ	anything
	On exit	$Cs	$Cs	character in
Notes:	$C8ØØ space is enabled. Pascal returns ACIA or keyboard data in the A Register and location $678+s with high bit cleared.			
$C9AA	Write			
	On entry	$Cs	$sØ	character out
	On exit	error code	$Cs	(changed)
Notes:	$C8ØØ space is enabled. Output character is transmitted through the ACIA. Pascal posts error code to IORESULT.			

Table A-1Ø. Pascal 1.Ø Entry Points Used by SSC

PASCAL 1.1 ENTRY POINTS

The Pascal 1.1 entry point protocol is outlined in the first section of this appendix. The values given here are the addresses of the routines. Unlike Pascal 1.Ø, Pascal 1.1 enters these routines using indirect addressing.

Addr.	Offset for	Value	X Register	Y Register	A Register
$CsØD	Initialization	$(Cs)8E			
	On entry		$Cs	$sØ	anything
	On exit		$ØØ	$sØ	(changed)
Notes:	$C8ØØ space is enabled. Firmware initializes SSC to default values plus SW1 and SW2 selections.				
$CsØE	Read	$(Cs)94			
	On entry		$Cs	$sØ	anything
	On exit		error code	$Cs	char. in
Notes:	$C8ØØ space is enabled. Character in from ACIA or keyboard is returned in the A Register.				
$CsØF	Write	$(Cn)97			
	On entry		$Cs	$sØ	char. out
	On exit		error code	$Cs	(changed)
Notes:	$C8ØØ space is enabled. The byte in the A Register is sent out through the ACIA.				
$CslØ	Status	$(Cs)9A			
	On entry		$Cs	$sØ	request (Ø or 1)
	On exit		error code	$sØ	error code
Notes:	$C8ØØ space is enabled. Request = Ø asks ACIA whether it is ready to transmit another byte; request = 1 asks ACIA whether it has an input character available. On exit, carry bit = Ø for Yes or 1 for No.				

Table A-11. Pascal 1.1 Offsets Used by SSC

OTHER SPECIAL FIRMWARE LOCATIONS

The SSC firmware uses several other addresses for predefined purposes. Table A-12 lists these locations.

Address	Value	Purpose
$CsØ5	$38	Pascal serial/firmware card identifier (as well as BASIC input entry point)
$CsØ7	$18	Pascal serial/firmware card identifier (as well as BASIC output entry point)
$CsØB	$Ø1	Pascal 1.1 generic signature byte ($Ø1 = firmware card)
$CsØC	$31	Pascal 1.1 Device Signature byte ($31 = serial or parallel I/O card #1)
$Cs11	$85	Pascal 1.1 optional routines flag (nonzero value = not supported)
$CsFF	$Ø8	Firmware revision level

Table A-12. SSC Special Firmware Locations

```
0000:            2 ********************************
0000:            3 *                              *
0000:            4 * APPLE II SSC FIRMWARE         *
0000:            5 *                              *
0000:            6 *   BY LARRY KENYON            *
0000:            7 *    -JANUARY 1981-            **************
0000:            8 *                                           *
0000:            9 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
0000:           10 *                                           *
0000:           11 ***********************************************
0000:           12 *                              *
0000:           13 * VARIABLE DEFINITIONS         *
0000:           14 *                              *
0000:           15 ********************************
0000:           16 ******************
0000:           17 * ZERO PAGE EQUS *
0000:           18 ******************
0024:           19 CH       EQU  $24        ;CURSOR HORIZONTAL POSITION
0026:           20 SLOT16   EQU  $26        ;SAVE $N0 TO FREE UP Y-REG
0027:           21 CHARACTER EQU $27        ;OUTPUT, SCREEN AND INPUT CHARS
0028:           22 BASL     EQU  $28        ;BASE SCREEN ADDRESS POINTER
0035:           23 ZPTEMP   EQU  $35        ;WORKHORSE TEMPORARY
002A:           24 ZPTMP1   EQU  $2A        ;WHEN ZPTEMP ISN'T ENOUGH
002B:           25 ZPTMP2   EQU  $2B        ;TEMPORARIES, TEMPORARIES!
0036:           26 CSWL     EQU  $36        ;CHAR OUT VECTOR
0037:           27 CSWH     EQU  $37
0038:           28 KSWL     EQU  $38        ;CHAR IN VECTOR
0039:           29 KSWH     EQU  $39
003C:           30 A1L      EQU  $3C        ;BATCH MOVE POINTER
004E:           31 RNDL     EQU  $4E        ;RANDOM NUMBER SEED
004F:           32 RNDH     EQU  $4F
0000:           33 ******************
0000:           34 * GENERAL EQUATES *
0000:           35 ******************
0100:           36 STACK    EQU  $100       ;SYSTEM STACK BLOCK
0200:           37 INBUFF   EQU  $200       ;SYSTEM INPUT BUFFER
C000:           38 KBD      EQU  $C000      ;KEYBOARD INPUT
C010:           39 KBDSTRB  EQU  $C010      ;KEYBOARD CLEAR
CFFF:           40 ROMSOFF  EQU  $CFFF      ;DISABLES C0-RES. $C800 ROMS
0000:           41 **********************
0000:           42 * SSC CARD ADDRESSES *
0000:           43 **********************
C081:           44 DIPSW1   EQU  $C081      ;(+$N0) DIPSWITCH BLOCK 1
C082:           45 DIPSW2   EQU  $C082      ;(+$N0) DIPSWITCH BLOCK 2
C088:           46 TDREG    EQU  $C088      ;(+$N0) TRANSMIT DATA REG (WRITE)
C088:           47 RDREG    EQU  $C088      ;(+$N0) READ DATA REG (READ)
C089:           48 STREG    EQU  $C089      ;(+$N0) STATUS REGISTER (READ)
C089:           49 RESET    EQU  $C089      ;(+$N0) SOFTWARE RESET (WRITE)
C08A:           50 CMDREG   EQU  $C08A      ;(+$N0) COMMAND REGISTER (R/W)
C08B:           51 CTLREG   EQU  $C08B      ;(+$N0) CONTROL REGISTER (R/W)
```

```
0000:          53 ************************************************************
0000:          54 *  BIT-> B7  B6  B5  B4  B3  B2  B1  B0
0000:          55 *        +---------------------------+
0000:          56 * DIPSW1 S1  S2  S3  S4  Z   Z   S5  S6  (LEFT DIPSWITCH)
0000:          57 *
0000:          58 * (S1-S4 USED FOR BAUD RATE, S5-S6 FOR FIRMWARE MODE)
0000:          59 *
0000:          60 * DIPSW2 S1  Z   S2  Z   S3  S4  S5  CTS (RIGHT DIPSWITCH)
0000:          61 *
0000:          62 * STREG  INT DSR DCD TDR RDR OVR FE  PE
0000:          63 *
0000:          64 * CTLREG STB << WL >> CK << BAUD RATE >>
0000:          65 *
0000:          66 * CMDREG <<PARITY >> ECH <<XMIT>> RE DTR
0000:          67 *
0000:          68 ************************************************************
0000:          69 ************************************
0000:          70 * SCREEN VARIABLES: PPC AND SIC MODES *
0000:          71 ************************************
0538:          72 CMDBYTE EQU  $5F8-$C0  ;HOLDS COMMAND CHARACTER (PPC & CIC)
0438:          73 HANDSHKE EQU $4F8-$C0  ;SIC P8A CHAR COUNTER FOR ETX/ACK
0438:          74 PARAMETER EQU $4F8-$C0 ;ACCUMULATOR FOR CMD PARAMETER
04B8:          75 STATEFLG EQU $578-$C0  ;
0000:          76 * B7=CR GEN ENB FLAG  B6=AFTER LC INPUT FLG
0000:          77 * B2-B0=COMMAND INTERPRETER STATES
0000:          78 *   0 0 0  IDLE
0000:          79 *   0 0 1  CMD CHAR RECEIVED
0000:          80 *   0 1 0  COLLECT <N> UNTIL CHAR THEN DO COMMAND
0000:          81 *   0 1 1  SKIP UNTIL SPACE, THEN GOTO STATE 4
0000:          82 *   1 0 0  E/D COMMANDS
0000:          83 *   1 0 1  UNUSED
0000:          84 *   1 1 0  WAIT UNTIL CR THEN SET STATE TO ZERO
0000:          85 *   1 1 1  WAIT UNTIL CR THEN DO PROC INDICATED BY PARM
0000:          86 *
0000:          87 * (B4-B0 DETERMINE ENQUIRE CHAR FOR P8A MODE)
0000:          88 *
03B8:          89 DELAYFLG EQU $478-$C0
0000:          90 * B7-B6=SCREEN TRANSLATION OPTIONS
0000:          91 *  0 0   LC->UC
0000:          92 *  0 1   NO TRANSLATION
0000:          93 *  1 0   LC->UC INVERSE
0000:          94 *  1 1   LC->UC, UC->UC INVERSE
0000:          95 *  (1-3 WILL ALLOW LC CHARS TO PASS THRU MONITOR)
0000:          96 *
0000:          97 * B5-B4=CR DELAY   0 0 = NO DELAY
0000:          98 * B3-B2=LF DELAY   0 1 = 32 MILLISEC
0000:          99 * B1-B0=FF DELAY   1 0 = 1/4 SEC
0000:         100 *                  1 1 = 2 SEC
0000:         101 *
05B8:         102 STSBYTE EQU  $678-$C0  ;STATUS/IORESULT/INPUT BYTE
0638:         103 PWDBYTE EQU  $6F8-$C0  ;PRINTER (FORMAT) WIDTH
06B8:         104 COLBYTE EQU  $778-$C0  ;COLUMN POSITION COUNTER
0738:         105 MISCFLG EQU  $7F8-$C0  ;
0000:         106 * B7=ECHO BIT         B6=TABBING OPTION ENABLE
0000:         107 * B5=LINEFEED EAT     B4=PASCAL/BASIC FLAG
0000:         108 * B3=XOFF ENB FLAG    B2=KEYBOARD ENB
0000:         109 * B1=PPC/CIC MODE     B0=LF GENERATE ENB
0000:         110 *
```

```
0000:           112 ****************************************
0000:           113 * TEMP SCREEN VARS (SLOT INDEPENDENT) *
0000:           114 ****************************************
07F8:           115 MSLOT   EQU  $7F8       ;BUFFER FOR HI SLOT ADDR ($CN)
0000:           116 *********************************
0000:           117 * SCREEN VARIABLES: CIC MODE     *
0000:           118 *********************************
0000:           119 *
0000:           120 * STATEFLG: B7=TERMINAL MODE FLAG
0000:           121 *         B3-B5=CHAIN SLOT
0000:           122 *
0638:           123 CHNBYTE EQU  $6F8-$C0  ;CURRENT OUTPUT SCREEN ($CN00 ENTRY)
0000:           124 *
0000:           125 * B0-B7=CN00 ENTRY
0000:           126 *
06B8:           127 BUFBYTE EQU  $778-$C0  ;BUFFER FOR ONE
0000:           128 *                       INPUT BYTE: HIGH BIT IS SET
0000:           129 *                       WHEN BUFFER IS FULL
0000:           130 *
0000:           131 * MISCFLG:             B6=TERM MODE SHIFT ENB
0000:           132 *
0000:           133 * OTHER SLOT VARIABLES AS DEFINED FOR PPC AND SIC MODES
0000:           134 *
0000:           135 ************************
0000:           136 * MONITOR SUBROUTINES *
0000:           137 ************************
FDED:           138 COUT    EQU  $FDED     ;CHARACTER OUT (THRU CSW)
FE89:           139 SETKBD  EQU  $FE89     ;SETS KSW TO APPLE KEYBOARD
FF58:           140 IORTS   EQU  $FF58     ;KNOWN "RTS" LOCATION
FCBA:           141 NXTA1   EQU  $FCBA     ;INCREMENT A1H,L AND CMP TO A2H,L
FE93:           142 SETSCR  EQU  $FE93     ;SETS CSW TO APPLE SCREEN
FDF6:           143 VIDOUT  EQU  $FDF6     ;OUTPUT A CHAR TO APPLE SCREEN
0000:           144         CHN  SSC.CN00
0000:             1 *********************************
0000:             2 *                               *
0000:             3 * APPLE II SSC FIRMWARE        *
0000:             4 *                               *
0000:             5 *   BY LARRY KENYON             *
0000:             6 *                               *
0000:             7 *    -JANUARY 1981-             **************
0000:             8 *                                           *
0000:             9 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
0000:            10 *                                           *
0000:            11 ***********************************************
0000:            12 *                               *
0000:            13 * CN00 SPACE CODE              *
0000:            14 *                               *
0000:            15 *********************************
----- NEXT OBJECT FILE NAME IS SSC.DCLS.OBJ0
C700:            16         ORG  $C700
C700:            17 *
C700:2C 58 FF    18 BINIT   BIT  IORTS     ;SET THE V-FLAG
C703:70 0C       19         BVS  BENTRY    ;<ALWAYS>
C705:38          20 IENTRY  SEC            ;BASIC INPUT ENTRY
C706:90          21         DFB  $90       ;OPCODE FOR BCC
C707:18          22 OENTRY  CLC            ;BASIC OUTPUT ENTRY
C708:B8          23         CLV
C709:50 06       24         BVC  BENTRY    ;<ALWAYS> SKIP AROUND PASCAL 1.1 ENTRY
```

```
C70B:01          25          DFB  $01        ;GENERIC SIGNATURE BYTE
C70C:31          26          DFB  $31        ;DEVICE SIGNATURE BYTE
C70D:8E          27          DFB  >PINIT
C70E:94          28          DFB  >PREAD
C70F:97          29          DFB  >PWRITE
C710:9A          30          DFB  >PSTATUS
C711:85 27       31 BENTRY   STA  CHARACTER
C713:86 35       32          STX  ZPTEMP     ;INPUT BUFFER INDEX
C715:8A          33          TXA              ;SAVE X AND Y REGS ON STACK
C716:48          34          PHA
C717:98          35          TYA
C718:48          36          PHA
C719:08          37          PHP              ;SAVE ENTRY FLAGS
C71A:78          38          SEI              ;NO RUPTS DURING SLOT DETERMINATION
C71B:8D FF CF    39          STA  ROMSOFF    ;SWITCH OUT OTHER $C800 ROMS
C71E:20 58 FF    40          JSR  IORTS
C721:BA          41          TSX
C722:BD 00 01    42          LDA  STACK,X    ;RECOVER $CN
C725:8D F8 07    43          STA  MSLOT
C728:AA          44          TAX              ;X-REG WILL GENERALLY BE $CN
C729:0A          45          ASL  A
C72A:0A          46          ASL  A          ;DETERMINE $N0
C72B:0A          47          ASL  A
C72C:0A          48          ASL  A
C72D:85 26       49          STA  SLOT16
C72F:A8          50          TAY              ;Y-REG WILL GENERALLY BE $N0
C730:28          51          PLP              ;RESTORE RUPTS
C731:50 29       52          BVC  NORMIO
C733:            53 *
C733:            54 * BASIC INITIALIZATION
C733:            55 *
C733:1E 38 05    56          ASL  CMDBYTE,X ;ALWAYS ENABLE COMMANDS
C736:5E 38 05    57          LSR  CMDBYTE,X
C739:B9 8A C0    58          LDA  CMDREG,Y  ;JUST HAD A POWER-ON OR PROGRAM RESET?
C73C:29 1F       59          AND  #$1F
C73E:D0 05       60          BNE  BINIT1
C740:A9 EF       61          LDA  #$EF       ;IF SO, GO JOIN INIT IN PROGRESS
C742:20 05 C8    62          JSR  INIT1
C745:            63 *
C745:E4 37       64 BINIT1   CPX  CSWH
C747:D0 0B       65          BNE  FROMIN
C749:A9 07       66          LDA  #>OENTRY
C74B:C5 36       67          CMP  CSWL       ;IF CSW IS ALREADY POINTING TO OENTRY,
C74D:F0 05       68          BEQ  FROMIN     ; THEN WE MUST HAVE COME FROM KSW
C74F:85 36       69          STA  CSWL       ;OTHERWISE, SET CSW TO OENTRY
C751:18          70 FROMOUT  CLC             ;INDICATE WE ARE CALLED FOR OUTPUT
C752:90 08       71          BCC  NORMIO     ;<ALWAYS>
C754:E4 39       72 FROMIN   CPX  KSWH       ;MAKE SURE KSW POINTS HERE
C756:D0 F9       73          BNE  FROMOUT    ;
C758:A9 05       74          LDA  #>IENTRY
C75A:85 38       75          STA  KSWL       ;SET UP KSW (NOTE CARRY SET FROM CPX)
C75C:            76 *
C75C:            77 * BRANCH TO APPROPRIATE BASIC I/O ROUTINE
C75C:            78 *
C75C:BD 38 07    79 NORMIO   LDA  MISCFLG,X ;SEPARATE CIC MODE FROM OTHERS
C75F:29 02       80          AND  #$02       ;NOT ZERO FOR CIC MODE
C761:08          81          PHP             ;SAVE CIC MODE INDICATION
C762:90 03       82          BCC  BOUTPUT
```

```
C764:4C BF C8   83          JMP  BINPUT
C767:           84 *
C767:BD B8 04   85 BOUTPUT LDA  STATEFLG,X ;CHECK FOR AFTER LOWERCASE INPUT
C76A:48         86          PHA
C76B:0A         87          ASL  A
C76C:10 0E      88          BPL  BOUTPUT1  ;SKIP IF NOT
C76E:A6 35      89          LDX  ZPTEMP
C770:A5 27      90          LDA  CHARACTER
C772:09 20      91          ORA  #$20
C774:9D 00 02   92          STA  INBUFF,X  ;RESTORE LOWERCASE IN BUFFER
C777:85 27      93          STA  CHARACTER ;AND FOR OUTPUT ECHO
C779:AE F8 07   94          LDX  MSLOT
C77C:68         95 BOUTPUT1 PLA
C77D:29 BF      96          AND  #$BF      ;ZERO THE FLAG
C77F:9D B8 04   97          STA  STATEFLG,X
C782:28         98          PLP            ;RETRIEVE CIC MODE INDICATION
C783:F0 06      99          BEQ  BOUTPUT2  ;BRANCH FOR PPC, SIC MODES
C785:20 63 CB  100          JSR  OUTPUT    ;CIC MODE OUTPUT
C788:4C B5 C8  101          JMP  CICEXIT   ;FINISH BY CHECKING FOR TERM MODE
C78B:          102 *
C78B:4C FC C8  103 BOUTPUT2 JMP SEROUT
C78E:          104 ********************************
C78E:          105 *                              *
C78E:          106 *  NEW PASCAL INTERFACE ENTRIES  *
C78E:          107 *                              *
C78E:          108 ********************************
C78E:20 00 C8  109 PINIT   JSR  PASCALINIT ;
C791:A2 00     110         LDX  #0         ;NO ERROR POSSIBLE
C793:60        111         RTS
C794:4C 9B C8  112 PREAD   JMP  PASCALREAD ;
C797:4C AA C9  113 PWRITE  JMP  PASCALWRITE ;
C79A:          114 *
C79A:          115 * NEW PASCAL STATUS REQUEST
C79A:          116 *
C79A:          117 * A-REG=0 -> READY FOR OUTPUT?
C79A:          118 * A-REG=1 -> HAS INPUT BEEN RECEIVED?
C79A:          119 *
C79A:4A        120 PSTATUS LSR  A          ;SAVE REQUEST TYPE IN CARRY
C79B:20 9B C9  121         JSR  PENTRY     ;(PRESERVES CARRY)
C79E:B0 08     122         BCS  PSTATIN
C7A0:20 F5 CA  123         JSR  SROUT      ;READY FOR OUTPUT?
C7A3:F0 06     124         BEQ  PSTATUS2
C7A5:18        125         CLC
C7A6:90 03     126         BCC  PSTATUS2  ;CARRY CLEAR FOR NOT READY
C7A8:          127 *
C7A8:20 D2 CA  128 PSTATIN JSR  SRIN       ;SETS CARRY CORRECTLY
C7AB:BD B8 05  129 PSTATUS2 LDA STSBYTE,X ;GET ERROR FLAGS
C7AE:AA        130         TAX
C7AF:60        131         RTS
C7B0:          132 ************************************************
C7B0:          133 * ROUTINE TO SEND A CHARACTER TO ANOTHER CARD *
C7B0:          134 ************************************************
C7B0:A2 03     135 SENDCD  LDX  #3
C7B2:B5 36     136 SAVEHOOK LDA CSWL,X
C7B4:48        137         PHA
C7B5:CA        138         DEX
C7B6:10 FA     139         BPL  SAVEHOOK
C7B8:          140 *
```

```
C7B8:           141 * NOW PUT CARD ADDRESS IN HOOK
C7B8:           142 *
C7B8:AE F8 07   143         LDX  MSLOT
C7BB:BD 38 06   144         LDA  CHNBYTE,X
C7BE:85 36      145         STA  CSWL
C7C0:BD B8 04   146         LDA  STATEFLG,X ;GET SLOT #
C7C3:29 38      147         AND  #$38
C7C5:4A         148         LSR  A
C7C6:4A         149         LSR  A
C7C7:4A         150         LSR  A
C7C8:09 C0      151         ORA  #$C0       ;FORM $CN
C7CA:85 37      152         STA  CSWH
C7CC:           153 *
C7CC:           154 * OUTPUT TO THE PERIPHERAL
C7CC:           155 *
C7CC:8A         156         TXA             ;SAVE $CN
C7CD:48         157         PHA
C7CE:A5 27      158         LDA  CHARACTER
C7D0:48         159         PHA
C7D1:09 80      160         ORA  #$80       ;80 COL BOARDS WANT HI-BIT ON
C7D3:20 ED FD   161         JSR  COUT
C7D6:           162 *
C7D6:           163 * NOW RESTORE EVERYTHING THE OTHER CARD MAY HAVE CLOBBERED
C7D6:           164 *
C7D6:68         165         PLA
C7D7:85 27      166         STA  CHARACTER
C7D9:68         167         PLA
C7DA:8D F8 07   168         STA  MSLOT
C7DD:AA         169         TAX
C7DE:0A         170         ASL  A
C7DF:0A         171         ASL  A
C7E0:0A         172         ASL  A
C7E1:0A         173         ASL  A
C7E2:85 26      174         STA  SLOT16
C7E4:8D FF CF   175         STA  ROMSOFF
C7E7:           176 *
C7E7:           177 * PUT BACK CSWL INTO CHNBYTE
C7E7:           178 *
C7E7:A5 36      179         LDA  CSWL
C7E9:9D 38 06   180         STA  CHNBYTE,X
C7EC:           181 *
C7EC:A2 00      182         LDX  #0
C7EE:68         183 RESTORHOOK PLA
C7EF:95 36      184         STA  CSWL,X
C7F1:E8         185         INX
C7F2:E0 04      186         CPX  #4
C7F4:90 F8      187         BCC  RESTORHOOK
C7F6:           188 *
C7F6:AE F8 07   189         LDX  MSLOT
C7F9:60         190         RTS
C7FA:           191 *
C7FA:C1 D0 D0   192         ASC  "APPLE"
C7FD:CC C5
C7FF:08         193         DFB  $8
C800:           194 *
```

```
C800:           196          CHN  SSC.C800
C800:             1 ********************************
C800:             2 *                              *
C800:             3 * APPLE II SSC FIRMWARE        *
C800:             4 *                              *
C800:             5 *   BY LARRY KENYON            *
C800:             6 *                              *
C800:             7 *    -JANUARY 1981-            **************
C800:             8 *                                           *
C800:             9 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
C800:            10 *                                           *
C800:            11 ***********************************************
C800:            12 *                              *
C800:            13 * C800 SPACE: HIGH LEVEL STUFF  *
C800:            14 *                              *
C800:            15 ********************************
C800:            16 * PASCAL 1.0 INIT ENTRY *
C800:            17 *************************
----- NEXT OBJECT FILE NAME IS SSC.DCLS.OBJ1
C800:            18          ORG  $C800
C800:20 9B C9    19 PASCALINIT JSR PENTRY  ;PASCAL 1.0 INITIALIZATION ENTRY
C803:A9 16       20          LDA  #$16     ;NO XOFF, ECHO, LF EAT, OR LF GEN
C805:48          21 INIT1    PHA           ;GOES TO MISCFLG AFTER MODIFICATION
C806:A9 00       22          LDA  #0
C808:9D B8 04    23          STA  STATEFLG,X
C80B:9D B8 03    24          STA  DELAYFLG,X
C80E:9D 38 04    25          STA  HANDSHKE,X
C811:9D B8 05    26          STA  STSBYTE,X
C814:9D 38 06    27          STA  PWDBYTE,X
C817:9D B8 06    28          STA  COLBYTE,X
C81A:B9 82 C0    29          LDA  DIPSW2,Y ;SET LF GEN OPTION FROM D2-S5
C81D:85 2B       30          STA  ZPTMP2   ;SAVE FOR LATER
C81F:4A          31          LSR  A        ;S5-> CARRY
C820:4A          32          LSR  A        ;IF S5=ON=0 THEN LEAVE MISCFLG ALONE
C821:90 04       33          BCC  INIT1A
C823:68          34          PLA           ;OTHERWISE, MAKE SURE LF GEN
C824:29 FE       35          AND  #$FE     ; ENABLE IS RESET
C826:48          36          PHA           ;
C827:B8          37 INIT1A   CLV           ;V WILL BE CLEAR FOR CIC MODE
C828:B9 81 C0    38          LDA  DIPSW1,Y
C82B:4A          39          LSR  A        ;SIC MODES SET CARRY
C82C:B0 07       40          BCS  INIT2    ;BRANCH FOR SIC MODES
C82E:4A          41          LSR  A
C82F:B0 0E       42          BCS  INIT2B   ;PPC MODE BRANCH
C831:A9 01       43          LDA  #$01     ;CTL-A
C833:D0 3D       44          BNE  INIT5    ;<ALWAYS> CIC MODE BRANCH
C835:            45 *
C835:4A          46 INIT2    LSR  A        ;SET CARRY FOR P8A
C836:A9 03       47          LDA  #$03     ;SET ETX AS DEFAULT INQUIRY CHAR
C838:B0 02       48          BCS  INIT2A   ;BRANCH FOR P8A
C83A:A9 80       49          LDA  #$80     ;FOR P8 SET AUTO CR GEN
C83C:9D B8 04    50 INIT2A   STA  STATEFLG,X
C83F:2C 58 FF    51 INIT2B   BIT  IORTS    ;SET V-FLAG FOR PPC, SIC MODES
C842:A5 2B       52          LDA  ZPTMP2
C844:29 20       53          AND  #$20     ;SET CR DELAY
C846:49 20       54          EOR  #$20     ;SO 1=ENB, 0=DISABLE
C848:9D B8 03    55          STA  DELAYFLG,X ; FROM D2-S2
C84B:            56 *
```

```
C84B:70 0A       57          BVS  INIT3      ;<ALWAYS> BRANCH AROUND PASCAL
C84D:            58 *************************°
C84D:            59 * PASCAL 1.0 READ ENTRY *
C84D:            60 * (MUST BE AT $C84D)    *
C84D:            61 *************************
C84D:20 9B C8    62 PREAD0   JSR  PASCALREAD ;DO PASCAL 1.1 READ
C850:AE F8 07    63          LDX  MSLOT      ;MODIFY FOR 1.0
C853:9D B8 05    64          STA  STSBYTE,X ;CHARACTER READ
C856:60          65          RTS
C857:            66 ************************
C857:            67 * NOW WHERE WERE WE??? *
C857:            68 ************************
C857:            69 *
C857:A5 2B       70 INIT3    LDA  ZPTMP2     ;PPC, SIC MODES USE SWITCHES
C859:4A          71          LSR  A          ; TO SET PWIDTH, CR DELAY
C85A:4A          72          LSR  A
C85B:29 03       73          AND  #$03
C85D:A8          74          TAY
C85E:F0 04       75          BEQ  INIT4
C860:            76 *
C860:68          77          PLA             ;RESET VIDEO ENABLE FOR PWIDTH#40
C861:29 7F       78          AND  #$7F
C863:48          79          PHA
C864:            80 *
C864:B9 A6 C9    81 INIT4    LDA  PWDTBL,Y
C867:9D 38 06    82          STA  PWDBYTE,X
C86A:A4 26       83          LDY  SLOT16
C86C:            84 *
C86C:68          85          PLA             ;CLEAR CIC BIT IN FUTURE MISCFLG
C86D:29 95       86          AND  #$95       ; (AND TABBING, XOFF  AND LF EAT BITS)
C86F:48          87          PHA
C870:A9 09       88          LDA  #$09       ;CTL-I
C872:            89 *
C872:9D 38 05    90 INIT5    STA  CMDBYTE,X ;CMD ESC CHAR (IGNORED FOR SIC MODES)
C875:68          91          PLA
C876:9D 38 07    92          STA  MISCFLG,X ;SET MISCFLG FLAGS
C879:            93 *
C879:            94 * NOW FOR THE ACIA INITIALIZATION  ROUTINE
C879:            95 *
C879:A5 2B       96 INITACIA LDA ZPTMP2     ;DIPSW2
C87B:48          97          PHA
C87C:29 A0       98          AND  #$A0       ;DATA BIT OPTIONS FOR CIC MODE
C87E:50 02       99          BVC  INITACIA1 ;BRANCH FOR CIC MODE
C880:29 80      100          AND  #$80       ;8 DATA, 1 OR 2 STOP FOR SIC, PPC
C882:20 A1 CD   101 INITACIA1 JSR DATACMD1 ;SET CONTROL REG
C885:20 81 CD   102          JSR  BAUDCMD1  ;SET DIPSWITCH BAUD RATE
C888:68         103          PLA
C889:29 0C      104          AND  #$0C       ;PARITY OPTIONS FOR CIC MODE
C88B:50 02      105          BVC  INITACIA2 ;BRANCH FOR CIC MODE
C88D:A9 00      106          LDA  #$0        ;DISABLE PARITY FOR SIC, PPC MODES
C88F:0A         107 INITACIA2 ASL A
C890:0A         108          ASL  A
C891:0A         109          ASL  A
C892:09 0B      110          ORA  #$0B
C894:99 8A C0   111          STA  CMDREG,Y
C897:B9 88 C0   112          LDA  RDREG,Y    ;THROW OUT THE STRANGE STUFF
C89A:60         113          RTS
C89B:           114 ***********************
```

```
C89B:           115 * PASCAL READ ROUTINE *
C89B:           116 ************************
C89B:20 9B C9   117 PASCALREAD JSR PENTRY  ;SHARED BY BOTH PASCAL VERSIONS
C89E:20 AA C8   118 PASCALREAD1 JSR GETCHAR ;GET ACIA/KBD DATA
C8A1:29 7F      119         AND  #$7F       ;CLEAR HIGH BIT FOR PASCAL
C8A3:AC F8 07   120 PASEXIT LDY  MSLOT
C8A6:BE B8 05   121         LDX  STSBYTE,Y ;ERROR STATUS-> X-REG
C8A9:60         122         RTS
C8AA:           123 ******************************
C8AA:           124 * GETCHAR ROUTINE WAITS FOR  *
C8AA:           125 *  THE NEXT CHAR FROM EITHER *
C8AA:           126 *  THE ACIA OR KEYBOARD (IF  *
C8AA:           127 *  ENABLED).  USED BY PASCAL *
C8AA:           128 *  READ ROUTINE, XON WAIT,   *
C8AA:           129 *  AND ACK WAIT. DATA IS RE- *
C8AA:           130 *  TURNED IN THE A-REGISTER  *
C8AA:           131 ******************************
C8AA:20 FF CA   132 GETCHAR JSR  INPUT      ;ACIA DATA?
C8AD:B0 05      133         BCS  GETCHAR1
C8AF:20 2C CC   134         JSR  CKKBD      ;KEYBOARD INPUT?
C8B2:90 F6      135         BCC  GETCHAR
C8B4:60         136 GETCHAR1 RTS            ;EXIT WHEN WE HAVE SOMETHING
C8B5:           137 *
C8B5:           138         CHN  SSC.HILEV
```

```
C8B5:            2 ********************************
C8B5:            3 *                              *
C8B5:            4 * APPLE II SSC FIRMWARE        *
C8B5:            5 *                              *
C8B5:            6 *   BY LARRY KENYON            *
C8B5:            7 *                              *
C8B5:            8 *    -FEBRUARY 1981-           **************
C8B5:            9 *                                           *
C8B5:           10 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
C8B5:           11 *                                           *
C8B5:           12 ************************************************
C8B5:           13 *                              *
C8B5:           14 * CIC, SIC, PPC MODE HIGH-LEVEL *
C8B5:           15 *                              *
C8B5:           16 ********************************
C8B5:           17 * CIC EXIT ROUTINE . . .        *
C8B5:           18 ********************************
C8B5:20 1E CA   19 CICEXIT JSR  CHECKTERM ;SEE IF WE'VE ENTERED TERMINAL MODE
C8B8:           20 **********************
C8B8:           21 * BASIC EXIT ROUTINE *
C8B8:           22 **********************
C8B8:68         23 BASICEXIT PLA
C8B9:A8         24          TAY
C8BA:68         25          PLA
C8BB:AA         26          TAX
C8BC:A5 27      27          LDA  CHARACTER
C8BE:60         28          RTS
C8BF:           29 ************************
C8BF:           30 * BASIC  INPUT ROUTINE *
C8BF:           31 ************************
C8BF:F0 29      32 BINPUT   BEQ  BINACIA    ;BRANCH IF NOT CIC MODE
C8C1:BD B8 06   33          LDA  BUFBYTE,X ;INPUT BUFFER FULL?
C8C4:10 05      34          BPL  BINKBD
C8C6:5E B8 06   35          LSR  BUFBYTE,X ;RESET BUFFER FULL
C8C9:D0 24      36          BNE  BINACIA1   ;<ALWAYS>
C8CB:           37 *
C8CB:20 3E CC   38 BINKBD   JSR  GETKBD     ;KEYBOARD DATA?
C8CE:90 1A      39          BCC  BINACIA
C8D0:           40 *
C8D0:BD B8 03   41 BINEND   LDA  DELAYFLG,X
C8D3:29 C0      42          AND  #$C0       ;TRANSLATE LOWERCASE TO UPPERCASE?
C8D5:F0 0E      43          BEQ  BINEND1    ;IF SO, LET THE MONITOR DO IT
C8D7:A5 27      44          LDA  CHARACTER ;IF NOT, SET FLAG IF
C8D9:C9 E0      45          CMP  #$E0       ; THIS IS A LOWERCASE CHAR
C8DB:90 08      46          BCC  BINEND1    ; FOR INPUT BUFFER CORRECTION
C8DD:BD B8 04   47          LDA  STATEFLG,X ; (CIRCUMVENT APPLE MONITOR)
C8E0:09 40      48          ORA  #$40
C8E2:9D B8 04   49          STA  STATEFLG,X
C8E5:           50 *
C8E5:28         51 BINEND1  PLP
C8E6:F0 D0      52          BEQ  BASICEXIT ;BRANCH IF NOT CIC MODE
C8E8:D0 CB      53          BNE  CICEXIT   ;<ALWAYS> CHECK TO SEE IF WE
C8EA:           54 *              ENTERED TERM MODE (VIA KYBD ESCAPE
C8EA:20 FF CA   55 BINACIA JSR  INPUT      ;ACIA DATA?
C8ED:90 DC      56          BCC  BINKBD
C8EF:20 11 CC   57 BINACIA1 JSR RESTORE    ;DO BASIC CURSED DUTY
C8F2:28         58          PLP
C8F3:08         59          PHP            ;GET CIC MODE INDICATOR
```

```
C8F4:F0 DA       60          BEQ  BINEND     ;SKIP IF NOT CIC MODE
C8F6:20 D1 C9    61          JSR  CKINPUT    ;LOOK FOR INPUT STREAM SPECIAL CHARS
C8F9:4C D0 C8    62          JMP  BINEND     ;
C8FC:            63 *********************************
C8FC:            64 * SIC, PPC BASIC OUTPUT ROUTINE *
C8FC:            65 *********************************
C8FC:20 1A CB    66 SEROUT   JSR  CMDSEQCK   ;CHECK FOR A COMMAND SEQUENCE
C8FF:B0 B7       67          BCS  BASICEXIT  ;BRANCH IF WE WERE IN COMMAND MODE
C901:A5 27       68          LDA  CHARACTER  ;SAVE CHAR ON STACK
C903:48          69          PHA
C904:BD 38 07    70          LDA  MISCFLG,X  ;IF VIDEO OR TABBING ENABLED,
C907:29 C0       71          AND  #$C0       ; DON'T MESS WITH THE CURSOR
C909:D0 16       72          BNE  TABCHECK
C90B:            73 *
C90B:A5 24       74          LDA  CH         ;CHECK FOR COMMA TABBING
C90D:F0 42       75          BEQ  NOTAB      ;IF CH=0, THERE WAS NO TAB OR COMMA
C90F:C9 08       76          CMP  #8         ;INTEGER BASIC COMMA?
C911:F0 04       77          BEQ  COMMA
C913:C9 10       78          CMP  #16        ;APPLESOFT COMMA?
C915:D0 0A       79          BNE  TABCHECK
C917:09 F0       80 COMMA    ORA  #$F0
C919:3D B8 06    81          AND  COLBYTE,X  ;SET COL TO PREVIOUS TAB
C91C:18          82          CLC
C91D:65 24       83          ADC  CH         ;THEN INCREMENT TO NEXT TAB
C91F:85 24       84          STA  CH
C921:            85 *
C921:            86 *
C921:BD B8 06    87 TABCHECK LDA  COLBYTE,X
C924:C5 24       88          CMP  CH         ;IS TABBING NEEDED?
C926:F0 29       89          BEQ  NOTAB      ;IF EQUAL THEN NO TAB NEEDED
C928:A9 A0       90          LDA  #$A0       ;SPACE FOR FORWARD TAB
C92A:90 08       91          BCC  TAB1
C92C:BD 38 07    92          LDA  MISCFLG,X  ;DON'T BACKSPACE UNLESS TABBING
C92F:0A          93          ASL  A          ; OPTION IS ENABLED
C930:10 1F       94          BPL  NOTAB
C932:A9 88       95          LDA  #$88       ;BACKSPACE FOR BACKTAB
C934:85 27       96 TAB1     STA  CHARACTER
C936:2C 58 FF    97          BIT  IORTS      ;SET V=1 TO INDICATE TABBING
C939:08          98          PHP             ;SAVE TABBING INDICATOR
C93A:70 0C       99          BVS  TAB2       ;<ALWAYS> AROUND BATCH MOVE ENTRY
C93C:EA         100          NOP
C93D:           101 *************************
C93D:           102 * SHORT BATCH MOVE:     *
C93D:           103 *  LOCATE AT $C93D FOR  *
C93D:           104 *  COMPATIBILITY WITH   *
C93D:           105 *  SIC P8 BLOCK MOVE.   *
C93D:           106 *************************
C93D:2C 58 FF   107 BATCHIN BIT  IORTS
C940:50         108          DFB  $50        ;DUMMY BVC
C941:B8         109 BATCHOUT CLV             ;V=0 FOR OUTPUT ENTRY
C942:AE F8 07   110          LDX  MSLOT
C945:4C EF C9   111          JMP  BATCHIO
C948:           112 ********************
C948:           113 * BURP . . .      *
C948:           114 ********************
C948:20 B5 C9   115 TAB2     JSR  ADJUST     ;ADJUST COLUMN COUNT
C94B:20 6B CB   116          JSR  OUTPUT2    ;DON'T GO TO SCREEN WHEN TABBING
C94E:4C 68 C9   117          JMP  FORCECR    ;SHARE SOME CODE. . .
```

```
C951:          118 *
C951:68        119 NOTAB    PLA
C952:B8        120          CLV
C953:08        121          PHP             ;SAVE 'NO TAB' INDICATION
C954:85 27     122 NOTAB1   STA  CHARACTER ;(FORCE CR REENTRY)
C956:48        123          PHA
C957:20 68 CB  124          JSR  OUTPUT1    ;ENTER AFTER CMD SEQ CHECK
C95A:20 B5 C9  125          JSR  ADJUST
C95D:68        126          PLA
C95E:49 8D     127          EOR  #$8D       ;WAS IT A CR?
C960:0A        128          ASL  A
C961:D0 05     129          BNE  FORCECR
C963:9D B8 06  130          STA  COLBYTE,X ;IF SO, RESET COLUMN TO 0
C966:85 24     131          STA  CH
C968:          132 *
C968:BD B8 04  133 FORCECR  LDA  STATEFLG,X ;FORCE CR DISABLED?
C96B:10 0D     134          BPL  SEREND
C96D:BD 38 06  135          LDA  PWDBYTE,X ;FORCE CR IF LIMIT REACHED
C970:F0 08     136          BEQ  SEREND     ;(FOR P8 POKE COMPATIBILITY)
C972:18        137          CLC
C973:FD B8 06  138          SBC  COLBYTE,X
C976:A9 8D     139          LDA  #$8D
C978:90 DA     140          BCC  NOTAB1     ;BRANCH TO FORCE CR
C97A:          141 *
C97A:28        142 SEREND   PLP
C97B:70 A4     143          BVS  TABCHECK   ;BRANCH IF TABBING
C97D:          144 *
C97D:BD 38 07  145          LDA  MISCFLG,X ;DON'T MESS WITH CURSOR
C980:30 16     146          BMI  SEREND2    ; WHEN VIDEO IS ON
C982:BC B8 06  147          LDY  COLBYTE,X
C985:0A        148          ASL  A
C986:30 0E     149          BMI  SETCH      ;SET CH TO VALUE OF COL FOR TABBING
C988:98        150          TYA
C989:A0 00     151          LDY  #0
C98B:38        152          SEC
C98C:FD 38 06  153          SBC  PWDBYTE,X ;
C98F:C9 F8     154          CMP  #$F8       ;WITHIN 8 CHARS OF PWIDTH?
C991:90 03     155          BCC  SETCH
C993:69 27     156          ADC  #$27       ;IF SO, ADJUST TO WITHIN 8 OF 40
C995:A8        157          TAY
C996:84 24     158 SETCH    STY  CH
C998:          159 *
C998:4C B8 C8  160 SEREND2  JMP  BASICEXIT ;THAT'S ALL
C99B:          161 *
C99B:          162 ****************************
C99B:          163 * PASCAL ENTRY ROUTINE     *
C99B:          164 ****************************
C99B:8E F8 07  165 PENTRY   STX  MSLOT
C99E:84 26     166          STY  SLOT16
C9A0:A9 00     167          LDA  #0
C9A2:9D B8 05  168          STA  STSBYTE,X
C9A5:60        169          RTS
C9A6:          170 *
C9A6:          171 ********************************
C9A6:          172 * SIC MODE PRINTER WIDTH TABLE *
C9A6:          173 ********************************
C9A6:29        174 PWDTBL   DFB  $29        ;40 COLUMNS
C9A7:48        175          DFB  $48        ;72 COLUMNS
```

```
C9A8:50          176          DFB  $50        ;80 COLUMNS
C9A9:84          177          DFB  $84        ;132 COLUMNS
C9AA:            178 ************************
C9AA:            179 * PASCAL WRITE ROUTINE *
C9AA:            180 * (DOUBLES AS PASCAL   *
C9AA:            181 *  1.0 ENTRY POINT)    *
C9AA:            182 *  -MUST BE AT $C9AA-  *
C9AA:            183 ************************
C9AA:85 27       184 PASCALWRITE STA CHARACTER
C9AC:20 9B C9    185          JSR  PENTRY
C9AF:20 63 CB    186          JSR  OUTPUT
C9B2:4C A3 C8    187          JMP  PASEXIT    ;LOAD X-REG WITH ERROR BYTE & RTS
C9B5:            188 *
C9B5:            189 ******************************
C9B5:            190 * COLUMN ADJUST ROUTINE      *
C9B5:            191 * (PPC, SIC MODES ONLY)      *
C9B5:            192 ******************************
C9B5:A5 27       193 ADJUST   LDA  CHARACTER
C9B7:49 08       194          EOR  #$08       ;BACKSPACE?
C9B9:0A          195          ASL  A
C9BA:F0 04       196          BEQ  DECRCOL    ;IF SO, DECREMENT COLUMN
C9BC:49 EE       197          EOR  #$EE       ;DELETE? ($FF, RUB)
C9BE:D0 09       198          BNE  CTRLTST
C9C0:DE B8 06    199 DECRCOL  DEC  COLBYTE,X ;DECREMENT COLUMN COUNT
C9C3:10 03       200          BPL  ADJRTS
C9C5:9D B8 06    201          STA  COLBYTE,X ;DON'T ALLOW TO GO BELOW 0
C9C8:60          202 ADJRTS   RTS
C9C9:C9 C0       203 CTRLTST  CMP  #$C0       ;DON'T INCREMENT COLUMN COUNT FOR
C9CB:B0 FB       204          BCS  ADJRTS     ; CONTROL CHARACTERS
C9CD:FE B8 06    205          INC  COLBYTE,X
C9D0:60          206          RTS
C9D1:            207 ****************************************
C9D1:            208 * ROUTINE TO PROCESS SPECIAL INPUT CHARS *
C9D1:            209 ****************************************
C9D1:BD 38 07    210 CKINPUT  LDA  MISCFLG,X
C9D4:29 08       211          AND  #$08       ;INPUT CTL CHARS ENABLED?
C9D6:F0 16       212          BEQ  CIEND
C9D8:            213 *
C9D8:BD B8 04    214          LDA  STATEFLG,X
C9DB:A4 27       215          LDY  CHARACTER
C9DD:C0 94       216          CPY  #$94       ;CTL-T?
C9DF:D0 04       217          BNE  CKINPUT1
C9E1:09 80       218          ORA  #$80       ;SET TERMINAL MODE
C9E3:D0 06       219          BNE  CKINPUT2   ;<ALWAYS>
C9E5:            220 *
C9E5:C0 92       221 CKINPUT1 CPY  #$92       ;CONTROL-R?
C9E7:D0 05       222          BNE  CIEND
C9E9:29 7F       223          AND  #$7F       ;RESET TERMINAL MODE
C9EB:9D B8 04    224 CKINPUT2 STA  STATEFLG,X
C9EE:60          225 CIEND    RTS
C9EF:            226 *
```

```
C9EF:           228         CHN  SSC.TERM
C9EF:             1 *********************************
C9EF:             2 *                               *
C9EF:             3 * APPLE II SSC FIRMWARE         *
C9EF:             4 *                               *
C9EF:             5 *   BY LARRY KENYON             *
C9EF:             6 *                               *
C9EF:             7 *     -APRIL 1981-              **************
C9EF:             8 *                                             *
C9EF:             9 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
C9EF:            10 *                                             *
C9EF:            11 ***************************************************
C9EF:            12 * SHORT BLOCK MOVE     *
C9EF:            13 ************************
C9EF:8A          14 BATCHIO TXA
C9F0:0A          15         ASL  A
C9F1:0A          16         ASL  A
C9F2:0A          17         ASL  A
C9F3:0A          18         ASL  A
C9F4:85 26       19         STA  SLOT16
C9F6:A9 00       20         LDA  #0
C9F8:9D B8 05    21         STA  STSBYTE,X ;ZERO ERROR INDICATION
C9FB:70 0F       22         BVS  MOVIN
C9FD:            23 *
C9FD:A0 00       24 MOVOUT  LDY  #0
C9FF:B1 3C       25         LDA  (A1L),Y   ;GET BUFFER DATA
CA01:85 27       26         STA  CHARACTER
CA03:20 02 CC    27         JSR  ACIAOUT   ;SEND IT OUT THE ACIA
CA06:20 BA FC    28         JSR  NXTA1
CA09:90 F2       29         BCC  MOVOUT
CA0B:60          30         RTS
CA0C:            31 *
CA0C:20 D2 CA    32 MOVIN   JSR  SRIN
CA0F:90 FB       33         BCC  MOVIN
CA11:B9 88 C0    34         LDA  RDREG,Y
CA14:A0 00       35         LDY  #0
CA16:91 3C       36         STA  (A1L),Y   ;PUT ACIA DATA INTO BUFFER
CA18:20 BA FC    37         JSR  NXTA1
CA1B:90 EF       38         BCC  MOVIN
CA1D:60          39         RTS
CA1E:            40 *
CA1E:            41 *********************************
CA1E:            42 *                                 *
CA1E:            43 *  TERMINAL MODE ROUTINES         *
CA1E:            44 *                                 *
CA1E:            45 *********************************
CA1E:BD B8 04    46 CHECKTERM LDA STATEFLG,X ;HAVE WE ENTERED TERMINAL MODE?
CA21:10 31       47         BPL  TERMRTS    ;IF NOT, A SIMPLE RTS WILL DO. . .
CA23:            48 *
CA23:            49 * WE ENTER THE WORLD OF TERMINAL MODE
CA23:            50 *
CA23:A9 02       51 TERMMODE LDA #$02       ;START IN SHIFT-LOCK STATE
CA25:48          52         PHA             ;SHIFT STATE IS SAVED ON STACK
CA26:A9 7F       53         LDA  #$7F
CA28:20 E2 CD    54         JSR  KCMD1      ;RESET ECHO (DEFAULT TO FULL DUP)
CA2B:            55 *
CA2B:A4 24       56 TERMNEXT LDY CH
CA2D:B1 28       57         LDA  (BASL),Y
```

```
CA2F:85 27      58          STA  CHARACTER ;SAVE SCREEN CHARACTER
CA31:A9 07      59 TERMNEXT1 LDA #$07     ;IMPLEMENT A FLASHING UNDERLINE
CA33:25 4F      60          AND  RNDH     ; FOR A CURSOR
CA35:D0 10      61          BNE  TERMNEXT3
CA37:A4 24      62          LDY  CH
CA39:A9 DF      63          LDA  #$DF
CA3B:D1 28      64          CMP  (BASL),Y ;IS UNDERLINE ON THE SCREEN?
CA3D:D0 02      65          BNE  TERMNEXT2 ;IF NOT, PUT IT THERE
CA3F:A5 27      66          LDA  CHARACTER ;OTHERWISE USE TRUE SCREEN CHAR
CA41:91 28      67 TERMNEXT2 STA (BASL),Y
CA43:E6 4F      68          INC  RNDH     ;MAKE IT FLASH, BUT
CA45:E6 4F      69          INC  RNDH     ;NOT TOO SLOW AND NOT TOO FAST
CA47:           70 *
CA47:BD B8 04   71 TERMNEXT3 LDA STATEFLG,X ;ARE WE STILL IN TERM MODE?
CA4A:30 09      72          BMI  TERMACIAIN ;IF SO, GO CHECK ACIA
CA4C:           73 *
CA4C:20 11 CC   74 TERMEXIT JSR RESTORE   ;ALWAYS REPLACE OUR CURSOR
CA4F:68         75          PLA           ;CLEAN UP THE STACK
CA50:A9 8D      76          LDA  #$8D     ;RETURN A <CR> TO COVER UP
CA52:85 27      77          STA  CHARACTER
CA54:60         78 TERMRTS RTS
CA55:           79 *
CA55:20 FF CA   80 TERMACIAIN JSR INPUT   ;ACIA INPUT?
CA58:90 0C      81          BCC  TERMKBDIN ;IF NOT, GO CHECK KEYBOARD
CA5A:20 11 CC   82          JSR  RESTORE  ;RESTORE CURSOR, INPUT->CHARACTER
CA5D:20 D1 C9   83          JSR  CKINPUT  ;CHECK FOR CTL-T, CTL-R
CA60:20 A3 CC   84          JSR  SCREENOUT1 ;INPUT->SCREEN ALWAYS
CA63:4C 2B CA   85          JMP  TERMNEXT ;
CA66:           86 *
CA66:20 3E CC   87 TERMKBDIN JSR GETKBD   ;KEYPRESS?
CA69:90 C6      88          BCC  TERMNEXT1 ;SKIP IF NOT
CA6B:70 BE      89          BVS  TERMNEXT ;BRANCH  IF WE DID A KBD ESCAPE SEQ.
CA6D:BD 38 07   90          LDA  MISCFLG,X ;SHIFTING ENABLED?
CA70:0A         91          ASL  A
CA71:10 22      92          BPL  TERMSEND1
CA73:68         93          PLA           ;RECOVER TERMSTATE
CA74:A8         94          TAY
CA75:A5 27      95          LDA  CHARACTER
CA77:C0 01      96          CPY  #1       ;1 = SHIFT LETTERS, XLATE NUMBERS
CA79:F0 20      97          BEQ  TERMCAP
CA7B:B0 34      98          BCS  TERMLOCK ;2 MEANS CAPS LOCK MODE
CA7D:           99 *
CA7D:C9 9B     100 TERMNORM CMP #$9B      ;ESC?
CA7F:D0 06     101          BNE  TERMLETTER
CA81:          102 *
CA81:C8        103 TERMINC INY            ;INCREMENT STATE
CA82:98        104 TERMINC1 TYA
CA83:48        105          PHA           ;PUT BACK ON STACK
CA84:4C 2B CA  106          JMP  TERMNEXT
CA87:          107 *
CA87:C9 C1     108 TERMLETTER CMP #$C1    ;<A?
CA89:90 08     109          BCC  TERMSEND
CA8B:C9 DB     110          CMP  #$DB     ;>Z?
CA8D:B0 04     111          BCS  TERMSEND
CA8F:09 20     112          ORA  #$20     ;IT'S A LETTER SO TRANSLATE TO LC
CA91:85 27     113          STA  CHARACTER
CA93:          114 *
CA93:98        115 TERMSEND TYA
```

```
CA94:48        116          PHA             ;PUT STATE BACK ON STACK
CA95:20 68 CB  117 TERMSEND1 JSR OUTPUT1    ;GO OUTPUT
CA98:4C 2B CA  118          JMP  TERMNEXT
CA9B:          119 *
CA9B:C9 9B     120 TERMCAP CMP  #$9B        ;TWO ESCAPES?
CA9D:F0 E2     121          BEQ  TERMINC
CA9F:C9 B0     122          CMP  #$B0       ;<0?
CAA1:90 0A     123          BCC  TERMCAP1
CAA3:C9 BB     124          CMP  #$BB       ;>COLON?
CAA5:B0 06     125          BCS  TERMCAP1
CAA7:          126 *
CAA7:          127 * ESC <NUMBER> SO TRANSLATE INTO MISSING ASCII CHAR
CAA7:          128 *
CAA7:A8        129          TAY
CAA8:B9 09 CA  130          LDA  TRANSLATE-$B0,Y
CAAB:85 27     131          STA  CHARACTER
CAAD:A0 00     132 TERMCAP1 LDY #0          ;BACK TO STATE 0
CAAF:F0 E2     133          BEQ  TERMSEND   ;<ALWAYS>
CAB1:          134 *
CAB1:C9 9B     135 TERMLQCK CMP #$9B        ;ESC?
CAB3:D0 DE     136          BNE  TERMSEND
CAB5:A0 00     137          LDY  #0
CAB7:F0 C9     138          BEQ  TERMINC1;<ALWAYS>
CAB9:          139 *
CAB9:          140 ****************************
CAB9:          141 * TRANSLATE TABLE          *
CAB9:          142 ****************************
CAB9:9B        143 TRANSLATE DFB $9B        ;ESC
CABA:9C        144          DFB  $9C        ;FS
CABB:9F        145          DFB  $9F        ;US
CABC:DB        146          DFB  $DB        ;LEFT BRACKET
CABD:DC        147          DFB  $DC        ;LEFT SLASH
CABE:DF        148          DFB  $DF        ;UNDERSCORE
CABF:FB        149          DFB  $FB        ;LEFT ENCLOSE
CAC0:FC        150          DFB  $FC        ;VERTICAL BAR
CAC1:FD        151          DFB  $FD        ;RIGHT ENCLOSE
CAC2:FE        152          DFB  $FE        ;TILDE
CAC3:FF        153          DFB  $FF        ;RUB
CAC4:          154 *
CAC4:          155          CHN  SSC.CORE
```

```
CAC4:            2 ********************************
CAC4:            3 *                              *
CAC4:            4 * APPLE II SSC FIRMWARE        *
CAC4:            5 *                              *
CAC4:            6 *   BY LARRY KENYON            *
CAC4:            7 *                              *
CAC4:            8 *    -JANUARY 1981-            **************
CAC4:            9 *                                           *
CAC4:           10 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
CAC4:           11 *                                           *
CAC4:           12 ************************************************
CAC4:           13 *                              *
CAC4:           14 * CORE SUBROUTINES             *
CAC4:           15 *                              *
CAC4:           16 ********************************
CAC4:           17 ********************************
CAC4:           18 * GENERAL PURPOSE WAIT ROUTINE  *
CAC4:           19 ********************************
CAC4:           20 *
CAC4:           21 * WAITMS WAITS FOR [A-REG] MILLISECONDS (256 IF A-REG=0)
CAC4:           22 *
CAC4:A2 CA      23 WAITMS  LDX  #202
CAC6:CA         24 WAITMS1 DEX            ;<DON'T LET THIS LOOP CROSS A PAGE>
CAC7:D0 FD      25         BNE  WAITMS1   ;5 MICROSECOND LOOP
CAC9:38         26         SEC
CACA:E9 01      27         SBC  #01
CACC:D0 F6      28         BNE  WAITMS
CACE:AE F8 07   29         LDX  MSLOT
CAD1:60         30         RTS
CAD2:           31 ************************************
CAD2:           32 * ACIA STATUS REGISTER READ ROUTINES *
CAD2:           33 ************************************
CAD2:           34 *
CAD2:           35 *   SRIN USED TO CHECK ACIA INPUT STATUS
CAD2:           36 *
CAD2:A4 26      37 SRIN    LDY  SLOT16    ;SLOT16=$N0
CAD4:B9 89 C0   38         LDA  STREG,Y
CAD7:48         39         PHA
CAD8:29 20      40         AND  #$20      ;DCD?
CADA:4A         41         LSR  A         ;AN ERROR IF NOT
CADB:4A         42         LSR  A
CADC:85 35      43         STA  ZPTEMP
CADE:68         44         PLA
CADF:29 0F      45         AND  #$0F
CAE1:C9 08      46         CMP  #$08      ;SET CARRY IF RDR FULL, ELSE CLEAR
CAE3:90 04      47         BCC  SRIN1
CAE5:29 07      48         AND  #$07      ;PE, FE, OVR VALID ONLY WHEN RDR=1
CAE7:B0 02      49         BCS  SRIN2     ;<ALWAYS>
CAE9:A5 35      50 SRIN1   LDA  ZPTEMP
CAEB:05 35      51 SRIN2   ORA  ZPTEMP    ;GET DCD ERROR BIT
CAED:F0 05      52         BEQ  SRIN3     ;BRANCH IF NO ERRORS FOUND
CAEF:09 20      53         ORA  #$20      ;ELSE SET BIT 5 TO OFFSET FOR PASCAL
CAF1:9D B8 05   54         STA  STSBYTE,X ;AND SAVE IN STATUS TEMP
CAF4:60         55 SRIN3   RTS            ;CY=1 MEANS DATA IS AVAILABLE
CAF5:           56 *
CAF5:           57 * SROUT CHECKS IF TDR IS EMPTY + HARDWARE HANDSHAKE IS OK
CAF5:           58 *
CAF5:A4 26      59 SROUT   LDY  SLOT16
```

```
CAF7:B9 89 C0    60          LDA  STREG,Y
CAFA:29 70       61          AND  #$70
CAFC:C9 10       62          CMP  #$10       ;EQU IF TDR EMPTY, DCD, DSR, & CTS
CAFE:60          63          RTS
CAFF:            64 *
CAFF:            65 *************************
CAFF:            66 * GENERAL INPUT ROUTINE *
CAFF:            67 *************************
CAFF:20 D2 CA    68 INPUT    JSR  SRIN
CB02:90 15       69          BCC  NOINPUT1
CB04:            70 *
CB04:B9 88 C0    71          LDA  RDREG,Y    ;GET THE ACIA INPUT
CB07:09 80       72          ORA  #$80       ;SET HI BIT FOR BASIC
CB09:C9 8A       73          CMP  #$8A       ;LINEFEED?
CB0B:D0 09       74          BNE  INPUT2
CB0D:            75 *
CB0D:A8          76          TAY
CB0E:BD 38 07    77          LDA  MISCFLG,X ;SEE IF WE SHOULD EAT IT
CB11:29 20       78          AND  #$20
CB13:D0 03       79          BNE  NOINPUT    ;IF SO, JUST KEEP IT A SECRET
CB15:98          80          TYA
CB16:            81 *
CB16:38          82 INPUT2   SEC             ;INDICATE DATA
CB17:60          83          RTS
CB18:            84 *
CB18:18          85 NOINPUT  CLC             ;CARRY CLEAR FOR NO INPUT
CB19:60          86 NOINPUT1 RTS
CB1A:            87 *
CB1A:            88 *************************
CB1A:            89 * GENERAL OUTPUT ROUTINE *
CB1A:            90 *************************
CB1A:            91 *
CB1A:            92 * START OF COMMAND CHECK ROUTINE
CB1A:            93 *
CB1A:A4 26       94 CMDSEQCK LDY SLOT16
CB1C:B9 81 C0    95          LDA  DIPSW1,Y
CB1F:4A          96          LSR  A
CB20:B0 36       97          BCS  NOCMD      ;DON'T WORRY ABOUT CMD SEQ FOR SIC
CB22:BD B8 04    98          LDA  STATEFLG,X
CB25:29 07       99          AND  #$07       ;ARE WE IN A COMMAND SEQUENCE?
CB27:F0 05      100          BEQ  ESCCHECK
CB29:20 FC CD   101          JSR  CMDPROC    ;IF SO, GOTO COMMAND CENTRAL
CB2C:38         102          SEC             ;INDICATE COMMAND
CB2D:60         103          RTS
CB2E:           104 *
CB2E:A5 27      105 ESCCHECK LDA CHARACTER
CB30:29 7F      106          AND  #$7F       ;IGNORE HIGH BIT
CB32:DD 38 05   107          CMP  CMDBYTE,X ;IS THIS BEGINNING OF A CMD SEQ?
CB35:D0 05      108          BNE  XOFFCK
CB37:FE B8 04   109          INC  STATEFLG,X ;START UP COMMAND MODES
CB3A:38         110          SEC             ;INDICATE COMMAND
CB3B:60         111          RTS
CB3C:           112 *
CB3C:BD 38 07   113 XOFFCK   LDA  MISCFLG,X ;IS XON ENABLED?
CB3F:29 08      114          AND  #$08
CB41:F0 15      115          BEQ  NOCMD      ;SKIP THIS IF NOT
CB43:           116 *
CB43:20 FF CA   117          JSR  INPUT      ;ANY INPUT?
```

```
CB46:90 10      118          BCC  NOCMD       ;IF NOT, GO OUTPUT
CB48:C9 93      119          CMP  #$93        ;IS IT AN XOFF?
CB4A:F0 0E      120          BEQ  XONWAIT     ;IF SO, GO WAIT FOR ANOTHER INPUT
CB4C:48         121          PHA
CB4D:BD 38 07   122          LDA  MISCFLG,X ;CIC MODE?
CB50:4A         123          LSR  A
CB51:4A         124          LSR  A
CB52:68         125          PLA
CB53:90 04      126          BCC  ANRTS
CB55:9D B8 06   127          STA  BUFBYTE,X ;IF SO, WE HAVE A BUFFER
CB58:18         128 NOCMD    CLC             ;INDICATE NOT A CMD SEQ
CB59:60         129 ANRTS    RTS
CB5A:           130 *
CB5A:20 AA C8   131 XONWAIT  JSR  GETCHAR    ;GET ACIA/KBD DATA
CB5D:C9 91      132          CMP  #$91        ;IS IT AN XON?
CB5F:D0 F9      133          BNE  XONWAIT     ;IF NOT, WAIT
CB61:18         134          CLC             ;OTHERWISE, INDICATE NOT A CMD SEQ
CB62:60         135          RTS             ; AND RETURN
CB63:           136 ************************************************
CB63:           137 * NOW THE OUTPUT ROUTINE YOU'VE BEEN WAITING FOR *
CB63:           138 ************************************************
CB63:20 1A CB   139 OUTPUT   JSR  CMDSEQCK
CB66:B0 F1      140          BCS  ANRTS      ;DON'T OUTPUT COMMAND SEQUENCES
CB68:           141 *
CB68:20 9E CC   142 OUTPUT1  JSR  SCREENOUT
CB6B:           143 *
CB6B:A4 26      144 OUTPUT2  LDY  SLOT16
CB6D:B9 81 C0   145          LDA  DIPSW1,Y
CB70:4A         146          LSR  A
CB71:90 4E      147          BCC  OUTPUT3    ;SKIP ETX/ACK FOR NATIVE MODES
CB73:4A         148          LSR  A
CB74:90 4B      149          BCC  OUTPUT3    ;BRANCH IF NOT P8A EMULATION
CB76:           150 *
CB76:           151 ********************
CB76:           152 * P8A ETX/ACK STUFF*
CB76:           153 ********************
CB76:           154 * AFTER 148 CHARACTERS BUT NOT WITHIN AN ESCAPE SEQUENCE
CB76:           155 * OF UP TO 5 CHARACTERS, THE HANDSHAKE IS PERFORMED
CB76:           156 * (WILL DELAY UNTIL 'NOT ESC' AND THEN 4 MORE CHARS
CB76:           157 *  OR UNTIL AN 'ESC')
CB76:           158 *
CB76:A5 27      159 P8AOUT1  LDA  CHARACTER ;SAVE CHAR ON STACK
CB78:48         160          PHA
CB79:BD 38 04   161          LDA  HANDSHKE,X ;CHAR COUNT FOR BUFFER FULL
CB7C:C9 67      162          CMP  #103       ;IF <103 THEN 153 CHARS IN BUFFER
CB7E:90 10      163          BCC  ETX
CB80:C9 6C      164          CMP  #108       ;IF >=108 THEN LESS THAN 149 CHARS
CB82:B0 22      165          BCS  P8AOUT2    ; SO NO HANDSHAKE IS NEEDED YET
CB84:C9 6B      166          CMP  #107       ;SETS CARRY IF 107 (149 SENT)
CB86:68         167          PLA
CB87:48         168          PHA
CB88:49 9B      169          EOR  #$9B       ;ESC?
CB8A:29 7F      170          AND  #$7F       ;IGNORE HI-BIT
CB8C:D0 18      171          BNE  P8AOUT2    ;COUNT AS 1 OF 5 IF NOT 'ESC'
CB8E:B0 19      172          BCS  P8AOUT3    ;DON'T COUNT IF 149TH CHAR IS 'ESC'
CB90:           173 *
CB90:BD B8 04   174 ETX      LDA  STATEFLG,X ;SEND QUERY CHAR TO PRINTER
CB93:29 1F      175          AND  #$1F       ;(DEFAULT IS ETX)
```

```
CB95:09 80      176          ORA  #$80
CB97:85 27      177          STA  CHARACTER
CB99:20 02 CC   178          JSR  ACIAOUT
CB9C:20 AA C8   179 ACK      JSR  GETCHAR    ;GET ACIA/KBD DATA
CB9F:49 86      180          EOR  #$86       ;ACK?
CBA1:D0 ED      181          BNE  ETX        ;IF NOT ACK, REPEAT HANDSHAKE
CBA3:9D 38 04   182          STA  HANDSHKE,X ;INIT CHAR COUNT TO 255
CBA6:           183 *
CBA6:DE 38 04   184 P8AOUT2  DEC  HANDSHKE,X
CBA9:68         185 P8AOUT3  PLA             ;GET REAL CHAR TO OUTPUT
CBAA:85 27      186          STA  CHARACTER
CBAC:49 8D      187          EOR  #$8D       ;IF CR AND CR DELAY MODE
CBAE:0A         188          ASL  A
CBAF:D0 0A      189          BNE  P8AOUT4    ; THEN FAKE CHAR COUNT TO LESS THAN
CBB1:BD B8 03   190          LDA  DELAYFLG,X ; 48 TO FORCE HANDSHAKE ON NEXT
CBB4:29 30      191          AND  #$30       ; CHARACTER OUT
CBB6:F0 03      192          BEQ  P8AOUT4
CBB8:9D 38 04   193          STA  HANDSHKE,X
CBBB:           194 *
CBBB:20 02 CC   195 P8AOUT4  JSR  ACIAOUT
CBBE:4C EA CB   196          JMP  LFGEN      ;(SKIP DELAYS)
CBC1:           197 ******************************
CBC1:           198 * AND BACK TO NORMAL OUTPUT *
CBC1:           199 ******************************
CBC1:20 02 CC   200 OUTPUT3  JSR  ACIAOUT    ;OUTPUT THE CHARACTER
CBC4:           201 *
CBC4:           202 * NOW CHECK FOR CR, LF, AND FF DELAYS
CBC4:           203 *
CBC4:0A         204          ASL  A
CBC5:A8         205          TAY
CBC6:BD B8 03   206          LDA  DELAYFLG,X ;GET DELAY FLAGS
CBC9:C0 18      207          CPY  #$18       ;FORM FEED?
CBCB:F0 0C      208          BEQ  OUTDLY1
CBCD:4A         209          LSR  A
CBCE:4A         210          LSR  A          ;RIGHT JUSTIFY LF DELAY
CBCF:C0 14      211          CPY  #$14       ;LINE FEED?
CBD1:F0 06      212          BEQ  OUTDLY1
CBD3:4A         213          LSR  A
CBD4:4A         214          LSR  A          ;RIGHT JUSTIFY CR DELAY
CBD5:C0 1A      215          CPY  #$1A       ;CARRIAGE RETURN?
CBD7:D0 25      216          BNE  OUTPUTEND
CBD9:29 03      217 OUTDLY1  AND  #$03       ;JUST WANT LOWEST 2 BITS
CBDB:F0 0D      218          BEQ  LFGEN      ;NO DELAY INDICATED
CBDD:A8         219          TAY
CBDE:B9 FE CB   220          LDA  DLYTBL-1,Y
CBE1:A8         221          TAY             ;DELAY IN 32 MSEC INCREMENTS
CBE2:A9 20      222 OUTDLYLP LDA #32         ;
CBE4:20 C4 CA   223          JSR  WAITMS
CBE7:88         224          DEY
CBE8:D0 F8      225          BNE  OUTDLYLP
CBEA:           226 *
CBEA:           227 * CHECK ON LF GENERATION OPTION
CBEA:           228 *
CBEA:A5 27      229 LFGEN    LDA  CHARACTER
CBEC:0A         230          ASL  A
CBED:C9 1A      231          CMP  #$1A       ;CARRIAGE RETURN?
CBEF:D0 0D      232          BNE  OUTPUTEND
CBF1:BD 38 07   233          LDA  MISCFLG,X ;IS LF GENERATE ENABLED?
```

```
CBF4:6A          234          ROR  A
CBF5:90 07       235          BCC  OUTPUTEND
CBF7:A9 8A       236          LDA  #$8A
CBF9:85 27       237          STA  CHARACTER ;LINE FEED
CBFB:4C 6B CB    238          JMP  OUTPUT2   ;(DON'T ECHO IT)
CBFE:60          239 OUTPUTEND RTS
CBFF:            240 *
CBFF:01          241 DLYTBL   DFB  $01       ;32 MSEC
CC00:08          242          DFB  $08       ;1/4 SEC
CC01:40          243          DFB  $40       ; 2  SEC
CC02:            244 *************************
CC02:            245 * ACIA OUTPUT ROUTINE   *
CC02:            246 *************************
CC02:20 F5 CA    247 ACIAOUT  JSR  SROUT     ;READY FOR OUTPUT?
CC05:D0 FB       248          BNE  ACIAOUT
CC07:98          249          TYA
CC08:09 89       250          ORA  #$89      ;PREPARE TO ADDRESS ACIA,
CC0A:A8          251          TAY            ; CAUSING 6502 FALSE READ TO OCCUR
CC0B:A5 27       252          LDA  CHARACTER ; ON PAGE $BF (AVOIDING RDR READ)
CC0D:99 FF BF    253          STA  $BFFF,Y   ;HERE YOU ARE ACIA
CC10:60          254          RTS
CC11:            255 *
CC11:            256 ***********************************
CC11:            257 * RESTORE CURSOR (NOT FOR PASCAL) *
CC11:            258 * (A-REG SHOULD CONTAIN NEW CHAR) *
CC11:            259 ***********************************
CC11:48          260 RESTORE  PHA            ;SAVE NEW CHARACTER
CC12:A4 24       261          LDY  CH
CC14:A5 27       262          LDA  CHARACTER ;OLD CHARACTER
CC16:91 28       263          STA  (BASL),Y
CC18:68          264          PLA
CC19:            265 *
CC19:C9 95       266          CMP  #$95      ;SCREEN PICK?
CC1B:D0 0C       267          BNE  RESTOREND
CC1D:A5 27       268          LDA  CHARACTER ;IF SO, USE SCREEN CHAR
CC1F:C9 20       269          CMP  #$20      ;INVERSE?
CC21:B0 06       270          BCS  RESTOREND
CC23:20 DF CC    271          JSR  GETXLATE  ;REVERSE THE TRANSLATION
CC26:59 DB CC    272          EOR  REVMASK,Y
CC29:85 27       273 RESTOREND STA CHARACTER
CC2B:60          274          RTS
CC2C:            275 *
CC2C:            276          CHN  SSC.UTIL
```

```
CC2C:            2 ********************************
CC2C:            3 *                              *
CC2C:            4 * APPLE II SSC FIRMWARE        *
CC2C:            5 *                              *
CC2C:            6 *   BY LARRY KENYON            *
CC2C:            7 *                              *
CC2C:            8 *    -JANUARY 1981-            **************
CC2C:            9 *                                           *
CC2C:           10 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
CC2C:           11 *                                           *
CC2C:           12 ************************************************
CC2C:           13 *                              *
CC2C:           14 * UTILITY ROUTINES             *
CC2C:           15 *                              *
CC2C:           16 ********************************
CC2C:           17 * PASCAL-BASIC KEYBOARD FETCH *
CC2C:           18 ******************************
CC2C:18         19 CKKBD   CLC             ;RETURN CARRY CLEAR FOR NO DATA
CC2D:BD 38 07   20         LDA  MISCFLG,X
CC30:29 04      21         AND  #$04       ;ANSWER NO IF KEYBOARD IS DISABLED
CC32:F0 09      22         BEQ  CKKBDXIT
CC34:           23 *
CC34:AD 00 C0   24 CKKBD1  LDA  KBD
CC37:10 04      25         BPL  CKKBDXIT
CC39:8D 10 C0   26         STA  KBDSTRB
CC3C:38         27         SEC             ;INDICATE DATA
CC3D:60         28 CKKBDXIT RTS
CC3E:           29 ****************************************
CC3E:           30 * GET A CHAR FROM KEYBOARD FOR BASIC ONLY  *
CC3E:           31 ****************************************
CC3E:E6 4E      32 GETKBD  INC  RNDL        ;MIX UP RANDOM # SEED
CC40:D0 02      33         BNE  GETKBD1     ; FOR BASIC
CC42:E6 4F      34         INC  RNDH
CC44:20 2C CC   35 GETKBD1 JSR  CKKBD       ;KEYBOARD FETCH ROUTINE
CC47:B8         36         CLV              ;INDICATE NO ESCAPE SEQUENCE
CC48:90 F3      37         BCC  CKKBDXIT    ;EXIT IF NO KEY PRESS
CC4A:20 11 CC   38         JSR  RESTORE     ;DO BASIC CURSED DUTY
CC4D:29 7F      39         AND  #$7F
CC4F:DD 38 05   40         CMP  CMDBYTE,X ;IS IT THE START OF A COMMAND?
CC52:D0 3D      41         BNE  GETKBDONE ;IF NOT, EXIT INDICATING DATA
CC54:A4 26      42         LDY  SLOT16
CC56:B9 81 C0   43         LDA  DIPSW1,Y  ;ONLY DO CMD ESC FOR PPC, SIC MODES
CC59:4A         44         LSR  A
CC5A:B0 35      45         BCS  GETKBDONE
CC5C:           46 ****************************
CC5C:           47 * KEYBOARD ESCAPE HANDLER *
CC5C:           48 ****************************
CC5C:A0 0A      49 KBDESC  LDY  #$A         ;FIRST PRINT A PROMPT
CC5E:B9 93 CC   50 PROMPTLOOP LDA PROMPTBL,Y
CC61:85 27      51         STA  CHARACTER
CC63:98         52         TYA
CC64:48         53         PHA
CC65:20 A3 CC   54         JSR  SCREENOUT1 ;ALWAYS SEND TO SCREEN
CC68:68         55         PLA
CC69:A8         56         TAY
CC6A:88         57         DEY
CC6B:10 F1      58         BPL  PROMPTLOOP
CC6D:           59 *
```

```
CC6D:A9 01      60          LDA  #1         ;START OUT IN COMMAND STATE 1
CC6F:20 7B CE   61          JSR  SETOSTATE
CC72:           62 *
CC72:20 34 CC   63 GETCMD   JSR  CKKBD1     ;WAIT FOR KEYBOARD CHARACTER
CC75:10 FB      64          BPL  GETCMD
CC77:C9 88      65          CMP  #$88       ;BACKSPACE?
CC79:F0 E1      66          BEQ  KBDESC     ;IF SO, THEN START OVER
CC7B:85 27      67          STA  CHARACTER
CC7D:           68 *
CC7D:20 A3 CC   69          JSR  SCREENOUT1
CC80:20 1A CB   70          JSR  CMDSEQCK   ;PUMP THRU CMD INTERPRETER
CC83:           71 *
CC83:BD B8 04   72          LDA  STATEFLG,X ;ARE WE DONE?
CC86:29 07      73          AND  #$07
CC88:D0 E8      74          BNE  GETCMD     ;IF NOT, GO AGAIN
CC8A:           75 *
CC8A:A9 8D      76          LDA  #$8D       ;FORCE BACK A CARRIAGE RETURN
CC8C:85 27      77          STA  CHARACTER
CC8E:2C 58 FF   78          BIT  IORTS      ;INDICATE THAT A CMD SEQ HAS OCCURRED
CC91:38         79 GETKBDONE SEC            ;INDICATE SUCCESS
CC92:60         80          RTS
CC93:           81 *
CC93:           82 *
CC93:BA C3 D3   83 PROMPTBL ASC ":CSS      ELPPA"
CC96:D3 A0 C5
CC99:CC D0 D0
CC9C:C1
CC9D:8D         84          DFB  $8D
CC9E:           85 *
CC9E:           86 ********************************************************
CC9E:           87 * ROUTINE TO PRINT A CHARACTER ON THE CURRENT DISPLAY *
CC9E:           88 ********************************************************
CC9E:BD 38 07   89 SCREENOUT LDA MISCFLG,X
CCA1:10 13      90          BPL  NOOUT      ;IF SCREEN DISABLED
CCA3:           91 *
CCA3:BD 38 07   92 SCREENOUT1 LDA MISCFLG,X ;ENTRY AFTER ECHO CHECK
CCA6:29 02      93          AND  #$02       ;IF IT ISN'T CIC MODE,
CCA8:F0 0D      94          BEQ  ASCREEN    ;ALWAYS USE THE APPLE SCREEN
CCAA:BD B8 04   95          LDA  STATEFLG,X ;CURRENT SCREEN = APPLE SCREEN?
CCAD:29 38      96          AND  #$38
CCAF:F0 06      97          BEQ  ASCREEN    ;SLOT 0= APPLE SCREEN
CCB1:           98 *
CCB1:8A         99          TXA             ;JUMP TO CN00 SPACE
CCB2:48        100          PHA
CCB3:A9 AF     101          LDA  #>SENDCD-1 ; TO VECTOR TO THE PERIPHERAL
CCB5:48        102          PHA             ; IN THE CHAIN SLOT
CCB6:60        103 NOOUT    RTS
CCB7:          104 *
CCB7:          105 * APPLE 40-COL SCREEN DRIVER
CCB7:          106 *
CCB7:20 DF CC  107 ASCREEN  JSR  GETXLATE   ;GET THE TRANSLATE OPTIONS
CCBA:09 80     108          ORA  #$80       ;SET HIGH BIT OF CHAR
CCBC:C9 E0     109          CMP  #$E0       ;LOWERCASE?
CCBE:90 06     110          BCC  TESTLETTER
CCC0:59 D3 CC  111          EOR  LCMASK,Y   ;DO LOWERCASE TRIP
CCC3:4C F6 FD  112 TOSCREEN JMP  VIDOUT     ;ALL REGS ARE PRESERVED
CCC6:          113 *
CCC6:          114 * IF UPPERCASE, WE ONLY MAP LETTERS
```

```
CCC6:           115 *
CCC6:C9 C1      116 TESTLETTER CMP #$C1     ;<A?
CCC8:90 F9      117            BCC  TOSCREEN
CCCA:C9 DB      118            CMP  #$DB    ;>Z?
CCCC:B0 F5      119            BCS  TOSCREEN
CCCE:59 D7 CC   120            EOR  UCMASK,Y
CCD1:90 F0      121            BCC  TOSCREEN  ;<ALWAYS>
CCD3:           122 *
CCD3:           123 * MASKS FOR CASE TRANSLATION
CCD3:20 00 E0   124 LCMASK   DFB  $20,$00,$E0,$20
CCD6:20
CCD7:00 00 00   125 UCMASK   DFB  $00,$00,$00,$C0
CCDA:C0
CCDB:00 00 E0   126 REVMASK DFB  $00,$00,$E0,$C0
CCDE:C0
CCDF:           127 *
CCDF:BD B8 03   128 GETXLATE LDA DELAYFLG,X ;TRANSLATE OPTIONS IN B6-B7
CCE2:2A         129            ROL  A
CCE3:2A         130            ROL  A
CCE4:2A         131            ROL  A
CCE5:29 03      132            AND  #$03
CCE7:A8         133            TAY
CCE8:A5 27      134            LDA  CHARACTER
CCEA:60         135            RTS
CCEB:           136 *
```

(listings continued on next page)

```
CCEB:          138         CHN  SSC.CMD
CCEB:            1 *********************************
CCEB:            2 *                               *
CCEB:            3 * APPLE II SSC FIRMWARE         *
CCEB:            4 *                               *
CCEB:            5 *   BY LARRY KENYON             *
CCEB:            6 *                               *
CCEB:            7 *    -JANUARY 1981-             **************
CCEB:            8 *                                           *
CCEB:            9 * (C) COPYRIGHT 1981 BY APPLE COMPUTER, INC. *
CCEB:           10 *                                           *
CCEB:           11 ***********************************************
CCEB:           12 *                               *
CCEB:           13 * SSC COMMAND PROCESSOR         *
CCEB:           14 *                               *
CCEB:           15 *********************************
CCEB:           16 ****************************************************
CCEB:           17 * COMMAND TABLE (USED BY COMMAND PROCESSER ROUTINE *
CCEB:           18 ****************************************************
CCEB:42         19 CMDTBL   DFB  $42        ;B(REAK)
CCEC:67         20          DFB  $67        ;CIC      PAS  NS=7
CCED:C0         21          DFB  >BREAKCMD-1
CCEE:54         22          DFB  $54        ;T(ERMINAL)
CCEF:47         23          DFB  $47        ;CIC           NS=7
CCF0:A6         24          DFB  >TERMCMD-1
CCF1:43         25          DFB  $43        ;C(R GENERATE)
CCF2:87         26          DFB  $87        ;     PPC      NS=7
CCF3:A6         27          DFB  >TERMCMD-1
CCF4:51         28          DFB  $51        ;Q(UIT)
CCF5:47         29          DFB  $47        ;CIC           NS=7
CCF6:B8         30          DFB  >QUITCMD-1
CCF7:52         31          DFB  $52        ;R(ESET)
CCF8:C7         32          DFB  $C7        ;CIC PPC       NS=7
CCF9:AC         33          DFB  >RESETCMD-1
CCFA:5A         34          DFB  $5A        ;Z COMMAND
CCFB:E7         35          DFB  $E7        ;CIC PPC PAS   NS=7
CCFC:F3         36          DFB  >ZCMD-1
CCFD:49         37          DFB  $49        ;I COMMAND
CCFE:90         38          DFB  $90        ;     PPC      NS=0
CCFF:D3         39          DFB  >ICMD-1
CD00:4B         40          DFB  $4B        ;K COMMAND
CD01:90         41          DFB  $90        ;     PPC      NS=0
CD02:DF         42          DFB  >KCMD-1
CD03:           43 *
CD03:45         44          DFB  $45        ;E(CHO)
CD04:43         45          DFB  $43        ;CIC           NS=3
CD05:80         46          DFB  $80
CD06:46         47          DFB  $46        ;F(ROMKYBD)
CD07:E3         48          DFB  $E3        ;CIC PPC PAS   NS=3
CD08:04         49          DFB  $04
CD09:4C         50          DFB  $4C        ;L(F GENERATE)
CD0A:E3         51          DFB  $E3        ;CIC PPC PAS   NS=3
CD0B:01         52          DFB  $01
CD0C:58         53          DFB  $58        ;X(OFF)
CD0D:E3         54          DFB  $E3        ;CIC PPC PAS   NS=3
CD0E:08         55          DFB  $08
CD0F:54         56          DFB  $54        ;T(ABBING)
CD10:83         57          DFB  $83        ;     PPC      NS=3
```

```
CD11:40          58          DFB  $40
CD12:53          59          DFB  $53        ;S(HIFTING)
CD13:43          60          DFB  $43        ;CIC            NS=3
CD14:40          61          DFB  $40
CD15:4D          62          DFB  $4D        ;M(UNCH LF)
CD16:E3          63          DFB  $E3        ;CIC PPC PAS    NS=3
CD17:20          64          DFB  $20
CD18:            65 *
CD18:00          66          DFB  $0C        ;END OF FIRST PART MARKER
CD19:            67 *
CD19:42          68 CMDTBL1  DFB  $42        ;B(AUD)
CD1A:F6          69          DFB  $F6        ;CIC PPC PAS    NS=6
CD1B:7C          70          DFB  >BAUDCMD-1
CD1C:50          71          DFB  $50        ;P(ARITY)
CD1D:F6          72          DFB  $F6        ;CIC PPC PAS    NS=6
CD1E:9A          73          DFB  >PARITYCMD-1
CD1F:44          74          DFB  $44        ;D(ATA)
CD20:F6          75          DFB  $F6        ;CIC PPC PAS    NS=6
CD21:9B          76          DFB  >DATACMD-1
CD22:46          77          DFB  $46        ;F(F DELAY)
CD23:F6          78          DFB  $F6        ;CIC PPC PAS    NS=6
CD24:46          79          DFB  >FFCMD-1
CD25:4C          80          DFB  $4C        ;L(F DELAY)
CD26:F6          81          DFB  $F6        ;CIC PPC PAS    NS=6
CD27:40          82          DFB  >LFCMD-1
CD28:43          83          DFB  $43        ;C(R DELAY)
CD29:F6          84          DFB  $F6        ;CIC PPC PAS    NS=6
CD2A:3A          85          DFB  >CRCMD-1
CD2B:54          86          DFB  $54        ;T(RANSLATE)
CD2C:D6          87          DFB  $D6        ;CIC PPC        NS=6
CD2D:34          88          DFB  >TRANCMD-1
CD2E:4E          89          DFB  $4E        ;N COMMAND
CD2F:90          90          DFB  $90        ;    PPC        NS=0
CD30:E8          91          DFB  >NCMD-1
CD31:53          92          DFB  $53        ;S(CREENSLOT)
CD32:56          93          DFB  $56        ;CIC            NS=6
CD33:60          94          DFB  >SSLOTCMD-1
CD34:            95 *
CD34:00          96          DFB  $00        ;END OF TABLE MARKER
CD35:            97 *
CD35:            98 **********************
CD35:            99 * COMMAND ROUTINES   *
CD35:           100 * (CALLED BY PARSER) *
CD35:           101 * (MUST START IN     *
CD35:           102 *  PAGE $CD . . . )  *
CD35:           103 **********************
CD35:A9 3F      104 TRANCMD LDA  #$3F       ;SET SCREEN TRANSLATE OPTIONS
CD37:A0 07      105         LDY  #$7
CD39:D0 10      106         BNE  DELAYSET   ;<ALWAYS>
CD3B:A9 CF      107 CRCMD   LDA  #$CF       ;SET CR DELAY
CD3D:A0 05      108         LDY  #$5
CD3F:D0 0A      109         BNE  DELAYSET   ;<ALWAYS>
CD41:           110 *
CD41:A9 F3      111 LFCMD   LDA  #$F3       ;SET LF DELAY
CD43:A0 03      112         LDY  #$3
CD45:D0 04      113         BNE  DELAYSET   ;<ALWAYS>
CD47:           114 *
CD47:A9 FC      115 FFCMD   LDA  #$FC       ;SET FF DELAY
```

```
CD49:A0 01      116          LDY  #$1
CD4B:3D B8 03   117 DELAYSET AND DELAYFLG,X ;DON'T DISTURB THE OTHER FLAGS
CD4E:85 2A      118          STA  ZPTMP1
CD50:BD 38 04   119          LDA  PARAMETER,X
CD53:29 03      120          AND  #$03       ;JUST USE TWO BITS
CD55:18         121          CLC
CD56:6A         122          ROR  A          ;ONCE FOR FUN
CD57:2A         123 ROTATE   ROL  A          ;CHANGE DIRECTIONS
CD58:88         124          DEY
CD59:D0 FC      125          BNE  ROTATE     ;PREPARE IT TO OR INTO THE FLAGS
CD5B:           126 *
CD5B:05 2A      127          ORA  ZPTMP1
CD5D:9D B8 03   128          STA  DELAYFLG,X
CD60:60         129          RTS
CD61:           130 *
CD61:29 07      131 SSLOTCMD AND #$7         ;SET SLOT COMMAND
CD63:0A         132          ASL  A
CD64:0A         133          ASL  A
CD65:0A         134          ASL  A
CD66:85 2A      135          STA  ZPTMP1
CD68:0A         136          ASL  A
CD69:C5 26      137          CMP  SLOT16     ;MAKE SURE WE DON'T SET IT
CD6B:F0 0F      138          BEQ  SSLOTCMD1  ; TO OUR OWN SLOT
CD6D:BD B8 04   139          LDA  STATEFLG,X
CD70:29 C7      140          AND  #$C7       ;PUT NEW SLOT NUMBER IN BITS 3-5
CD72:05 2A      141          ORA  ZPTMP1     ; OF CMDBYTE,X
CD74:9D B8 04   142          STA  STATEFLG,X
CD77:A9 00      143          LDA  #0         ;STORE ZERO INTO
CD79:9D 38 06   144          STA  CHNBYTE,X ;SLOT OFFSET (SET TO CN00 ENTRY)
CD7C:60         145 SSLOTCMD1 RTS
CD7D:           146 *
CD7D:29 0F      147 BAUDCMD  AND  #$0F       ;SET NEW BAUD RATE
CD7F:D0 07      148          BNE  BAUDCMD2
CD81:B9 81 C0   149 BAUDCMD1 LDA  DIPSW1,Y  ;ZERO PARM = RELOAD FROM SWITCHES
CD84:4A         150          LSR  A
CD85:4A         151          LSR  A
CD86:4A         152          LSR  A
CD87:4A         153          LSR  A
CD88:09 10      154 BAUDCMD2 ORA  #$10       ;SET INT. BAUD RATE GENERATOR
CD8A:85 2A      155          STA  ZPTMP1
CD8C:A9 E0      156          LDA  #$E0
CD8E:85 2B      157 CTLREGSET STA ZPTMP2
CD90:B9 8B C0   158          LDA  CTLREG,Y
CD93:25 2B      159          AND  ZPTMP2
CD95:05 2A      160          ORA  ZPTMP1
CD97:99 8B C0   161          STA  CTLREG,Y
CD9A:60         162          RTS
CD9B:           163 *
CD9B:88         164 PARITYCMD DEY            ;TRICK: SO CTLREG,Y ACTUALLY
CD9C:           165 *                        ADDRESSES THE COMMAND REG
CD9C:           166 *
CD9C:0A         167 DATACMD  ASL  A          ;SET NEW # OF DATA BITS
CD9D:0A         168          ASL  A
CD9E:0A         169          ASL  A
CD9F:0A         170          ASL  A
CDA0:0A         171          ASL  A
CDA1:85 2A      172 DATACMD1 STA  ZPTMP1
CDA3:A9 1F      173          LDA  #$1F
```

```
CDA5:D0 E7      174          BNE  CTLREGSET ;<ALWAYS>
CDA7:           175 *
CDA7:1E B8 04   176 TERMCMD  ASL  STATEFLG,X ;SET TERMINAL MODE
CDAA:38         177          SEC
CDAB:B0 10      178          BCS  QCMD1     ;<ALWAYS>
CDAD:           179 *
CDAD:99 89 C0   180 RESETCMD STA  RESET,Y   ;DROP RTS, DTR
CDB0:20 93 FE   181          JSR  SETSCR    ;PR#0
CDB3:20 89 FE   182          JSR  SETKBD    ;IN#0
CDB6:AE F8 07   183          LDX  MSLOT
CDB9:1E B8 04   184 QUITCMD  ASL  STATEFLG,X ;CLEAR TERMINAL MODE
CDBC:18         185          CLC
CDBD:7E B8 04   186 QCMD1    ROR  STATEFLG,X
CDC0:60         187          RTS
CDC1:           188 *
CDC1:B9 8A C0   189 BREAKCMD LDA  CMDREG,Y  ;SEND BREAK SIGNAL
CDC4:48         190          PHA             ; FOR 233 MILLISECONDS
CDC5:09 0C      191          ORA  #$0C
CDC7:99 8A C0   192          STA  CMDREG,Y
CDCA:A9 E9      193          LDA  #233      ;DELAY FOR 233 MICROSEC.
CDCC:20 C4 CA   194          JSR  WAITMS
CDCF:68         195          PLA             ;RESTORE OLD COMMAND REG CONTENTS
CDD0:99 8A C0   196          STA  CMDREG,Y
CDD3:60         197          RTS
CDD4:           198 *
CDD4:A9 28      199 ICMD     LDA  #$28
CDD6:9D 38 06   200          STA  PWDBYTE,X ;SET PRINTER WIDTH TO 40
CDD9:A9 80      201          LDA  #$80
CDDB:1D 38 07   202          ORA  MISCFLG,X ;SET SCREEN ECHO
CDDE:D0 05      203          BNE  KCMD2     ;<ALWAYS>
CDE0:           204 *
CDE0:A9 FE      205 KCMD     LDA  #$FE      ;RESET THE LF GENERATE FLAG
CDE2:3D 38 07   206 KCMD1    AND  MISCFLG,X
CDE5:9D 38 07   207 KCMD2    STA  MISCFLG,X
CDE8:60         208          RTS
CDE9:           209 *
CDE9:C9 28      210 NCMD     CMP  #40       ;>=40?
CDEB:90 0E      211          BCC  ZCMDRTS   ;IF NOT, JUST EXIT
CDED:9D 38 06   212          STA  PWDBYTE,X ;SET NEW PRINTER WIDTH
CDF0:A9 3F      213          LDA  #$3F      ;DISABLE SCREEN, SET LISTING MODE
CDF2:D0 EE      214          BNE  KCMD1     ;<ALWAYS>
CDF4:           215 *
CDF4:1E 38 05   216 ZCMD     ASL  CMDBYTE,X ;DISABLE COMMAND RECOGNITION
CDF7:38         217          SEC
CDF8:7E 38 05   218          ROR  CMDBYTE,X
CDFB:60         219 ZCMDRTS  RTS
CDFC:           220 *
CDFC:           221 ************************************
CDFC:           222 * VECTOR ACCORDING TO COMMAND STATE *
CDFC:           223 ************************************
CDFC:A8         224 CMDPROC  TAY             ;A-REG=COMMAND STATE
CDFD:A5 27      225          LDA  CHARACTER
CDFF:29 7F      226          AND  #$7F
CE01:           227 *
CE01:C9 20      228          CMP  #$20       ;SKIP SPACES FOR ALL MODES
CE03:D0 09      229          BNE  CMDPROC2
CE05:C0 03      230          CPY  #$3        ;EXCEPT MODE 3
CE07:F0 01      231          BEQ  CMDPROC1
```

```
CE09:60         232         RTS
CE0A:A9 04      233 CMDPROC1 LDA #$4
CE0C:D0 6D      234         BNE  SETOSTATE ;<ALWAYS>
CE0E:           235 *
CE0E:C9 0D      236 CMDPROC2 CMP #$0D      ;CARRIAGE RETURN?
CE10:D0 12      237         BNE  CMDPROC4  ;
CE12:20 79 CE   238         JSR  ZEROSTATE ;ABORT FOR STATES 0-5, EXIT FOR 6,7
CE15:C0 07      239         CPY  #$07      ;IN STATE 7 WE VECTOR TO THE PROC
CE17:F0 01      240         BEQ  CMDPROC3  ;
CE19:60         241         RTS            ;OTHERWISE, JUST EXIT
CE1A:           242 *
CE1A:A9 CD      243 CMDPROC3 LDA #$CD      ;ALL PROCS MUST START IN PAGE $CD
CE1C:48         244         PHA
CE1D:BD 38 04   245         LDA  PARAMETER,X
CE20:48         246         PHA
CE21:A4 26      247         LDY  SLOT16    ;NEEDED BY BREAK CMD
CE23:60         248         RTS
CE24:           249 *
CE24:85 35      250 CMDPROC4 STA ZPTEMP
CE26:A9 CE      251         LDA  #$CE      ;ALL ROUTINES MUST START
CE28:48         252         PHA            ; IN PAGE $CE
CE29:B9 30 CE   253         LDA  STATETBL,Y
CE2C:48         254         PHA
CE2D:A5 35      255         LDA  ZPTEMP
CE2F:60         256         RTS            ;RTS TO COMMAND PROCEDURE
CE30:           257 *
CE30:           258 * NOW THE STATE ROUTINES
CE30:           259 *
CE30:           260 ************************
CE30:           261 *  STATE BRANCH TABLE *
CE30:           262 ************************
CE30:A7         263 STATETBL DFB >STATERR-1 ;BAD STATE
CE31:37         264         DFB  >CSTATE1-1 ;<CMD> SEEN
CE32:61         265         DFB  >CSTATE2-1 ;ACCUMULATE PARAMETER
CE33:89         266         DFB  >CDONE-1   ;SKIP UNTIL SPACE
CE34:8A         267         DFB  >CSTATE4-1 ;E/D SOMETHING
CE35:A7         268         DFB  >STATERR-1 ;ILLEGAL STATE
CE36:89         269         DFB  >CDONE-1   ;SKIP UNTIL CR
CE37:89         270         DFB  >CDONE-1   ;SKIP UNTIL CR THEN DO CMD
CE38:           271 ********************
CE38:           272 * COMMAND STATE 1 *
CE38:           273 ********************
CE38:DD 38 05   274 CSTATE1 CMP  CMDBYTE,X ;IS IT <CMD>?
CE3B:D0 06      275         BNE  CSTATE1A
CE3D:DE B8 04   276         DEC  STATEFLG,X ;SET STATE BACK TO ZERO
CE40:4C 02 CC   277         JMP  ACIAOUT    ;OUTPUT <CMD> IF SO
CE43:           278 *
CE43:C9 30      279 CSTATE1A CMP #$30      ;>=0?
CE45:90 0D      280         BCC  CSTATE1B
CE47:C9 3A      281         CMP  #$3A      ;<=9?
CE49:B0 09      282         BCS  CSTATE1B
CE4B:29 0F      283         AND  #$0F      ;IT'S A NUMBER
CE4D:9D 38 04   284         STA  PARAMETER,X
CE50:A9 02      285         LDA  #2
CE52:D0 27      286         BNE  SETOSTATE ;<ALWAYS> SET MODE 2 AND RETURN
CE54:           287 *
CE54:C9 20      288 CSTATE1B CMP #$20      ;IS IT A CONTROL CHAR?
CE56:B0 06      289         BCS  CSTATE1C
```

```
CE58:9D 38 05  290          STA  CMDBYTE,X ;SET NEW COMMAND CHARACTER
CE5B:4C 79 CE  291          JMP  ZEROSTATE ;RESET STATE TO ZERO
CE5E:          292 *
CE5E:A0 00     293 CSTATE1C LDY #0         ;USE COMMAND TABLE
CE60:F0 4D     294          BEQ  CMDSEARCH ;<ALWAYS>
CE62:          295 ****************************************
CE62:          296 * COMMAND STATE 2: ACCUMULATE PARAMETER *
CE62:          297 ****************************************
CE62:49 30     298 CSTATE2 EOR  #$30       ;CONVERT $30-$39 TO 0-9
CE64:C9 0A     299          CMP  #$A        ;0-9?
CE66:B0 0D     300          BCS  CSTATE2A
CE68:A0 0A     301          LDY  #$A        ;IT'S A NUMBER, SO ADD
CE6A:7D 38 04  302 ACCLOOP ADC  PARAMETER,X ; IT TO 10*PARAMETER
CE6D:88        303          DEY
CE6E:D0 FA     304          BNE  ACCLOOP
CE70:9D 38 04  305          STA  PARAMETER,X
CE73:F0 15     306          BEQ  CDONE      ;<ALWAYS>
CE75:          307 *
CE75:A0 2E     308 CSTATE2A LDY #CMDTBL1-CMDTBL ;USE COMMAND TABLE
CE77:D0 36     309          BNE  CMDSEARCH ;<ALWAYS>
CE79:          310 **********************
CE79:          311 * SET COMMAND STATE *
CE79:          312 **********************
CE79:A9 00     313 ZEROSTATE LDA #0
CE7B:85 2A     314 SETOSTATE STA ZPTMP1
CE7D:AE F8 07  315          LDX  MSLOT
CE80:BD B8 04  316          LDA  STATEFLG,X
CE83:29 F8     317          AND  #$F8
CE85:05 2A     318          ORA  ZPTMP1
CE87:9D B8 04  319          STA  STATEFLG,X
CE8A:60        320 CDONE    RTS
CE8B:          321 ***************************
CE8B:          322 * COMMAND STATE 4 (E/D) *
CE8B:          323 ***************************
CE8B:A8        324 CSTATE4 TAY             ;E/D -> Y-REG
CE8C:BD 38 04  325          LDA  PARAMETER,X
CE8F:C0 44     326          CPY  #$44       ;D(ISABLE)?
CE91:F0 09     327          BEQ  CSTATE4A
CE93:C0 45     328          CPY  #$45       ;E(NABLE)?
CE95:D0 11     329          BNE  STATERR    ;IF NOT, IGNORE THIS COMMAND
CE97:1D 38 07  330          ORA  MISCFLG,X ;SET FLAG
CE9A:D0 05     331          BNE  CSTATE4B  ;<ALWAYS>
CE9C:49 FF     332 CSTATE4A EOR #$FF       ;INVERT FOR DISABLE
CE9E:3D 38 07  333          AND  MISCFLG,X ;RESET FLAG
CEA1:9D 38 07  334 CSTATE4B STA MISCFLG,X
CEA4:          335 **********************
CEA4:          336 * ESCAPE TO STATE 6 *
CEA4:          337 **********************
CEA4:A9 06     338 SETSTATE6 LDA #6
CEA6:D0 D3     339          BNE  SETOSTATE ;<ALWAYS>
CEA8:A9 20     340 STATERR LDA  #32        ;CODE FOR BAD COMMAND
CEAA:9D B8 05  341          STA  STSBYTE,X
CEAD:D0 F5     342          BNE  SETSTATE6 ;<ALWAYS>
CEAF:          343 ***********************************
CEAF:          344 * TABLE DRIVEN COMMAND PROCESSOR *
CEAF:          345 ***********************************
CEAF:B9 EB CC  346 CMDSEARCH LDA CMDTBL,Y ;GET CANDIDATE CHARACTER
CEB2:F0 F4     347          BEQ  STATERR   ;A ZERO MARKS THE END OF A SUBTABLE
```

```
CEB4:C5 35      348          CMP  ZPTEMP     ;MATCH?
CEB6:F0 05      349          BEQ  CMDMATCH
CEB8:C8         350          INY
CEB9:C8         351 CMDSEARCH1 INY           ;REENTRY FOR WRONG MODES
CEBA:C8         352          INY             ;ENTRY LENGTH = 3
CEBB:D0 F2      353          BNE  CMDSEARCH  ;<ALWAYS>
CEBD:           354 *
CEBD:C8         355 CMDMATCH INY
CEBE:B9 EB CC   356          LDA  CMDTBL,Y
CEC1:85 2A      357          STA  ZPTMP1
CEC3:29 20      358          AND  #$20       ;CHECK PASCAL ENABLE
CEC5:D0 07      359          BNE  CMDMATCH1  ;IT'S ON SO DONT CHECK P-BIT
CEC7:BD 38 07   360          LDA  MISCFLG,X  ;OFF SO MAKE SURE
CECA:29 10      361          AND  #$10       ;  THAT WE AREN'T IN PASCAL
CECC:D0 EB      362          BNE  CMDSEARCH1 ;BRANCH IF WE ARE
CECE:           363 *
CECE:BD 38 07   364 CMDMATCH1 LDA MISCFLG,X ;GET CIC/PPC BIT
CED1:4A         365          LSR  A          ;SHIFT CIC/PPC MODE BIT TO CARRY
CED2:4A         366          LSR  A
CED3:24 2A      367          BIT  ZPTMP1     ;PPC->N CIC->V
CED5:B0 04      368          BCS  CMDMATCH2  ;BRANCH IF CIC MODE
CED7:10 E0      369          BPL  CMDSEARCH1 ;NOT OK FOR PPC
CED9:30 02      370          BMI  CMDEXEC    ;AND OK
CEDB:50 DC      371 CMDMATCH2 BVC CMDSEARCH1 ;NOT OK FOR CIC
CEDD:           372 *
CEDD:A5 2A      373 CMDEXEC LDA  ZPTMP1     ;RETRIEVE TABLE MODE BYTE
CEDF:48         374          PHA
CEE0:29 07      375          AND  #$07
CEE2:20 7B CE   376          JSR  SETOSTATE ;SET NEXT STATE
CEE5:C8         377          INY
CEE6:68         378          PLA
CEE7:29 10      379          AND  #$10       ;
CEE9:D0 07      380          BNE  CMDEXEC1   ;IF BIT 4 IS SET, VECTOR TO ROUTINE
CEEB:B9 EB CC   381          LDA  CMDTBL,Y
CEEE:9D 38 04   382          STA  PARAMETER,X
CEF1:60         383          RTS
CEF2:           384 *
CEF2:A9 CD      385 CMDEXEC1 LDA #$CD       ;ROUTINES MUST BE IN PAGE $CD
CEF4:48         386          PHA
CEF5:B9 EB CC   387          LDA  CMDTBL,Y
CEF8:48         388          PHA
CEF9:A4 26      389          LDY  SLOT16
CEFB:BD 38 04   390          LDA  PARAMETER,X ;LOT OF ROUTINES NEED THIS
CEFE:60         391          RTS
CEFF:           392 *
CEFF:00         393          DFB  $00

SYMBOL TABLE    SORTED BY SYMBOL

   3C A1L          CE6A ACCLOOP       CC02 ACIAOUT       ?CB9C ACK
 C9C8 ADJRTS       C9B5 ADJUST        CB59 ANRTS          CCB7 ASCREEN
 C8B8 BASICEXIT      28 BASL         ?C93D BATCHIN        C9EF BATCHIO
?C941 BATCHOUT     CD7D BAUDCMD        CD81 BAUDCMD1       CD88 BAUDCMD2
 C711 BENTRY       C8EF BINACIA1       C8EA BINACIA        C8E5 BINEND1
 C8D0 BINEND       C745 BINIT1        ?C700 BINIT          C8CB BINKBD
 C8BF BINPUT       C77C BOUTPUT1       C767 BOUTPUT        C78B BOUTPUT2
 CDC1 BREAKCMD     06B8 BUFBYTE        CE8A CDONE            24 CH
   27 CHARACTER    CA1E CHECKTERM      0638 CHNBYTE        C8B5 CICEXIT
 C9EE CIEND        C9D1 CKINPUT        C9E5 CKINPUT1       C9EB CKINPUT2
```

```
CC3D CKKBDXIT       CC2C CKKBD          CC34 CKKBD1         0538 CMDBYTE
CEF2 CMDEXEC1       CEDD CMDEXEC        CECE CMDMATCH1      CEBD CMDMATCH
CEDB CMDMATCH2      CE0A CMDPROC1       CE0E CMDPROC2       CE1A CMDPROC3
CE24 CMDPROC4       CDFC CMDPROC        C08A CMDREG         CEAF CMDSEARCH
CEB9 CMDSEARCH1     CB1A CMDSEQCK       CD19 CMDTBL1        CCEB CMDTBL
06B8 COLBYTE        C917 COMMA          FDED COUT           CD3B CRCMD
CE43 CSTATE1A       CE54 CSTATE1B       CE5E CSTATE1C       CE38 CSTATE1
CE75 CSTATE2A       CE62 CSTATE2        CE9C CSTATE4A       CEA1 CSTATE4B
CE8B CSTATE4          37 CSWH             36 CSWL           C08B CTLREG
CD8E CTLREGSET      C9C9 CTRLTST        CDA1 DATACMD1       CD9C DATACMD
C9C0 DECRCOL        03B8 DELAYFLG       CD4B DELAYSET       C081 DIPSW1
C082 DIPSW2         CBFF DLYTBL         CB2E ESCCHECK       C890 ETX
CD47 FFCMD          C968 FORCECR        C754 FROMIN         C751 FROMOUT
C8B4 GETCHAR1       C8AA GETCHAR        CC72 GETCMD         CC3E GETKBD
CC44 GETKBD1        CC91 GETKBDONE      CCDF GETXLATE       0438 HANDSHKE
CDD4 ICMD           C705 IENTRY         0200 INBUFF         C805 INIT1
C827 INIT1A         C835 INIT2          C83C INIT2A         C83F INIT2B
C857 INIT3          C864 INIT4          C872 INIT5         ?C879 INITACIA
C882 INITACIA1      C88F INITACIA2      CB16 INPUT2         CAFF INPUT
FF58 IORTS          C010 KBDSTRB        C000 KBD            CC5C KBDESC
CDE2 KCMD1          CDE5 KCMD2          CDE0 KCMD             39 KSWH
  38 KSWL           CCD3 LCMASK         CD41 LFCMD          CBEA LFGEN
0738 MISCFLG        CA0C MOVIN          C9FD MOVOUT         07F8 MSLOT
CDE9 NCMD           CB58 NOCMD          CB19 NOINPUT1       CB18 NOINPUT
CCB6 NOOUT          C75C NORMIO         C954 NOTAB1         C951 NOTAB
FCBA NXTA1          C707 OENTRY         CBD9 OUTDLY1        CBE2 OUTDLYLP
CB68 OUTPUT1        CB6B OUTPUT2        CB63 OUTPUT         CBC1 OUTPUT3
CBFE OUTPUTEND     ?CB76 P8AOUT1        CBA6 P8AOUT2        CBA9 P8AOUT3
CBBB P8AOUT4        0438 PARAMETER      CD9B PARITYCMD      C800 PASCALINIT
?C89E PASCALREAD1   C89B PASCALREAD     C9AA PASCALWRITE    C8A3 PASEXIT
C99B PENTRY         C78E PINIT         ?C84D PREAD0         C794 PREAD
CC93 PROMPTBL       CC5E PROMPTLOOP     C7A8 PSTATIN        C79A PSTATUS
C7AB PSTATUS2       0638 PWDBYTE        C9A6 PWDTBL         C797 PWRITE
CDBD QCMD1          CDB9 QUITCMD        C088 RDREG          C089 RESET
CDAD RESETCMD       CC11 RESTORE        CC29 RESTOREND      C7EE RESTORHOOK
CCDB REVMASK          4F RNDH             4E RNDL           CFFF ROMSOFF
CD57 ROTATE         C7B2 SAVEHOOK       CC9E SCREENOUT      CCA3 SCREENOUT1
C7B0 SENDCD         C998 SEREND2        C97A SEREND         C8FC SEROUT
C996 SETCH          FE89 SETKBD         CE7B SETOSTATE      FE93 SETSCR
CEA4 SETSTATE6        26 SLOT16         CAE9 SRIN1          CAEB SRIN2
CAD2 SRIN           CAF4 SRIN3          CAF5 SROUT          CD7C SSLOTCMD1
CD61 SSLOTCMD       0100 STACK          04B8 STATEFLG       CEA8 STATERR
CE30 STATETBL       C089 STREG          05B8 STSBYTE        C934 TAB1
C948 TAB2           C921 TABCHECK      ?C088 TDREG          CA55 TERMACIAIN
CAAD TERMCAP1       CA9B TERMCAP        CDA7 TERMCMD       ?CA4C TERMEXIT
CA82 TERMINC1       CA81 TERMINC        CA66 TERMKBDIN      CA87 TERMLETTER
CAB1 TERMLOCK      ?CA23 TERMMODE       CA2B TERMNEXT       CA31 TERMNEXT1
CA41 TERMNEXT2      CA47 TERMNEXT3     ?CA7D TERMNORM       CA54 TERMRTS
CA93 TERMSEND       CA95 TERMSEND1      CCC6 TESTLETTER     CCC3 TOSCREEN
CD35 TRANCMD        CAB9 TRANSLATE      CCD7 UCMASK         FDF6 VIDOUT
CAC4 WAITMS         CAC6 WAITMS1        CB3C XOFFCK         CB5A XONWAIT
CDFB ZCMDRTS        CDF4 ZCMD           CE79 ZEROSTATE        35 ZPTEMP
  2A ZPTMP1           2B ZPTMP2


SYMBOL TABLE    SORTED BY ADDRESS

  24 CH               26 SLOT16           27 CHARACTER        28 BASL
  2A ZPTMP1           2B ZPTMP2           35 ZPTEMP           36 CSWL
  37 CSWH             38 KSWL             39 KSWH             3C A1L
  4E RNDL             4F RNDH           0100 STACK          0200 INBUFF
03B8 DELAYFLG       0438 HANDSHKE       0438 PARAMETER      04B8 STATEFLG
```

```
 0538 CMDBYTE        05B8 STSBYTE        0638 PWDBYTE        0638 CHNBYTE
 06B8 COLBYTE        06B8 BUFBYTE        0738 MISCFLG        07F8 MSLOT
 C000 KBD            C010 KBDSTRB        C081 DIPSW1         C082 DIPSW2
?C088 TDREG          C088 RDREG          C089 STREG          C089 RESET
 C08A CMDREG         C08B CTLREG        ?C700 BINIT          C705 IENTRY
 C707 OENTRY         C711 BENTRY         C745 BINIT1         C751 FROMOUT
 C754 FROMIN         C75C NORMIO         C767 BOUTPUT        C77C BOUTPUT1
 C78B BOUTPUT2       C78E PINIT          C794 PREAD          C797 PWRITE
 C79A PSTATUS        C7A8 PSTATIN        C7AB PSTATUS2       C7B0 SENDCD
 C7B2 SAVEHOOK       C7EE RESTORHOOK     C800 PASCALINIT     C805 INIT1
 C827 INIT1A         C835 INIT2          C83C INIT2A         C83F INIT2B
?C84D PREAD0         C857 INIT3          C864 INIT4          C872 INIT5
?C879 INITACIA       C882 INITACIA1      C88F INITACIA2      C89B PASCALREAD
?C89E PASCALREAD1    C8A3 PASEXIT        C8AA GETCHAR        C8B4 GETCHAR1
 C8B5 CICEXIT        C8B8 BASICEXIT      C8BF BINPUT         C8CB BINKBD
 C8D0 BINEND         C8E5 BINEND1        C8EA BINACIA        C8EF BINACIA1
 C8FC SEROUT         C917 COMMA          C921 TABCHECK       C934 TAB1
?C93D BATCHIN       ?C941 BATCHOUT       C948 TAB2           C951 NOTAB
 C954 NOTAB1         C968 FORCECR        C97A SEREND         C996 SETCH
 C998 SEREND2        C99B PENTRY         C9A6 PWDTBL         C9AA PASCALWRITE
 C9B5 ADJUST         C9C0 DECRCOL        C9C8 ADJRTS         C9C9 CTRLTST
 C9D1 CKINPUT        C9E5 CKINPUT1       C9EB CKINPUT2       C9EE CIEND
 C9EF BATCHIO        C9FD MOVOUT         CA0C MOVIN          CA1E CHECKTERM
?CA23 TERMMODE       CA2B TERMNEXT       CA31 TERMNEXT1      CA41 TERMNEXT2
 CA47 TERMNEXT3     ?CA4C TERMEXIT       CA54 TERMRTS        CA55 TERMACIAIN
 CA66 TERMKBDIN     ?CA7D TERMNORM       CA81 TERMINC        CA82 TERMINC1
 CA87 TERMLETTER     CA93 TERMSEND       CA95 TERMSEND1      CA9B TERMCAP
 CAAD TERMCAP1       CAB1 TERMLOCK       CAB9 TRANSLATE      CAC4 WAITMS
 CAC6 WAITMS1        CAD2 SRIN           CAE9 SRIN1          CAEB SRIN2
 CAF4 SRIN3          CAF5 SROUT          CAFF INPUT          CB16 INPUT2
 CB18 NOINPUT        CB19 NOINPUT1       CB1A CMDSEQCK       CB2E ESCCHECK
 CB3C XOFFCK         CB58 NOCMD          CB59 ANRTS          CB5A XONWAIT
 CB63 OUTPUT         CB68 OUTPUT1        CB6B OUTPUT2       ?CB76 P8AOUT1
 CB90 ETX           ?CB9C ACK            CBA6 P8AOUT2        CBA9 P8AOUT3
 CBBB P8AOUT4        CBC1 OUTPUT3        CBD9 OUTDLY1        CBE2 OUTDLYLP
 CBEA LFGEN          CBFE OUTPUTEND      CBFF DLYTBL         CC02 ACIAOUT
 CC11 RESTORE        CC29 RESTOREND      CC2C CKKBD          CC34 CKKBD1
 CC3D CKKBDXIT       CC3E GETKBD         CC44 GETKBD1        CC5C KBDESC
 CC5E PROMPTLOOP     CC72 GETCMD         CC91 GETKBDONE      CC93 PROMPTBL
 CC9E SCREENOUT      CCA3 SCREENOUT1     CCB6 NOOUT          CCB7 ASCREEN
 CCC3 TOSCREEN       CCC6 TESTLETTER     CCD3 LCMASK         CCD7 UCMASK
 CCDB REVMASK        CCDF GETXLATE       CCEB CMDTBL         CD19 CMDTBL1
 CD35 TRANCMD        CD3B CRCMD          CD41 LFCMD          CD47 FFCMD
 CD4B DELAYSET       CD57 ROTATE         CD61 SSLOTCMD       CD7C SSLOTCMD1
 CD7D BAUDCMD        CD81 BAUDCMD1       CD88 BAUDCMD2       CD8E CTLREGSET
 CD9B PARITYCMD      CD9C DATACMD        CDA1 DATACMD1       CDA7 TERMCMD
 CDAD RESETCMD       CDB9 QUITCMD        CDBD QCMD1          CDC1 BREAKCMD
 CDD4 ICMD           CDE0 KCMD           CDE2 KCMD1          CDE5 KCMD2
 CDE9 NCMD           CDF4 ZCMD           CDFB ZCMDRTS        CDFC CMDPROC
 CE0A CMDPROC1       CE0E CMDPROC2       CE1A CMDPROC3       CE24 CMDPROC4
 CE30 STATETBL       CE38 CSTATE1        CE43 CSTATE1A       CE54 CSTATE1B
 CE5E CSTATE1C       CE62 CSTATE2        CE6A ACCLOOP        CE75 CSTATE2A
 CE79 ZEROSTATE      CE7B SETOSTATE      CE8A CDONE          CE8B CSTATE4
 CE9C CSTATE4A       CEA1 CSTATE4B       CEA4 SETSTATE6      CEA8 STATERR
 CEAF CMDSEARCH      CEB9 CMDSEARCH1     CEBD CMDMATCH       CECE CMDMATCH1
 CEDB CMDMATCH2      CEDD CMDEXEC        CEF2 CMDEXEC1       CFFF ROMSOFF
 FCBA NXTA1          FDED COUT           FDF6 VIDOUT         FE89 SETKBD
 FE93 SETSCR         FF58 IORTS
```

APPENDIX B
APPLE INTERFACE CARD EMULATION

The SSC emulates both the P8 and the P8A versions of the Apple II Serial Interface Card (SIC), although the SSC is not completely POKE-compatible with either. In addition, the SSC supports several Apple II Communications Card and Parallel Card software commands.

OLD SERIAL INTERFACE CARD EMULATION

The SSC replaces the P8 and P8A versions of the Apple II Serial Interface Card (SIC) and it has two switch-selectable modes to emulate them, as explained below. However, because of firmware space limitations, the SSC does not support all functions of the older interface cards, and various POKE locations are different. This section explains these functional differences.

It is best to use Printer Mode rather than one of the emulation modes, except under these circumstances:

- if you have extensive existent applications that use PEEKs and POKEs to modify SIC operating characteristics
- if you need SIC P8A mode's ETX/ACK (or other-character/ACK) handshaking capabilities

What the SSC does NOT support that the old SIC does:

- P8 SIC block moves
- baud rates other than the 15 listed in the various baud rate tables in this manual (ACIA hardware generates only those 15)
- data formats other than 5 - 8 data bits and 1, 1-1/2 or 2 stop bits (ACIA characteristic; other formats rarely used anyway)
- <ESC>U and <ESC>L commands for upper and lowercase (but SSC's Translate command offers more options; POKEs also available)
- current-loop operation

To run the SSC in emulation of the old Apple II Serial Interface Card (SIC), prepare and install the SSC the same way as for Printer Mode (Chapters 1 and 2), with the following exceptions:

- Set mode switches SW1-5 ON and SW1-6 OFF to emulate the old SIC with a P8 ROM.
- Set mode switches SW1-5 OFF and SW1-6 OFF to emulate the old SIC with a P8A ROM.
- Install the SSC in whatever slot the old SIC was installed in for the application involved.
- Follow the instructions given in the next sections if the application program did PEEKs and POKEs.

P8 EMULATION POKES

Changing SIC parameters was done either by setting the seven switches located on the card, or by POKEing the SIC slot RAM locations where this configuration data was stored. BASIC programs that talked through the old SIC may be used with the new SSC; however, if the program POKEs at these slot RAM locations, those POKEs must be changed to be compatible with the SSC's use of the RAM. The P8 and P8A ROMs differ slightly in their use of these RAM locations. Tables B-1 and B-2 show the transformation for P8 mode; additional differences for P8A mode are noted in the following section. Other POKE possibilities are described in Appendix A.

In the tables, the letter s stands for the slot number (1-7) in which the SSC is installed; the other letters are used as variables whose values are noted in the table (sometimes further down).

There is no claim that making these changes is simple. In fact, whenever possible it is best to use Printer Mode and its software commands to change SSC operating variables.

Here is an example of how to use the tables: let's say that the SSC is in slot #3. You want: a baud rate of 11Ø; data format of 5 data bits and 2 stop bits, even parity; line width of 4Ø with video on, no automatic <LF> after <CR>; no translation of lowercase to uppercase; and no 1/4-second delay after <CR>. The PEEKs and POKEs:

```
POKE 49339, 243    (49291 + 3*16; 3 + 24Ø)
POKE 49338, 1Ø7    (4929Ø + 3*16; p = 1Ø7)
POKE 2Ø43, 132     (plug in magic number)
POKE 1147, 64      (plug in magic number)
```

The same thing in Printer Mode with appropriate switch settings is:

```
SW1-1 to SW1-7: ON  ON  OFF OFF OFF ON  ON
SW2-1 to SW2-7: --  OFF ON  ON  OFF OFF OFF
```

Then to set 5 data and 2 stop bits, use <CTRL-I>7D<RETURN>; for even parity, use <CTRL-I>3P<RETURN>; to leave lowercase alone, use <CTRL-I>1T<RETURN>. You can use commands to change baud rate, etc.

Selection	SSC switches and settings	PEEKs and POKES to use for P8 Serial Card	PEEKs and POKES to use for Super Serial Card
P8 Mode: P8A Mode:	SW1-5 ON, SW1-6 OFF SW1-5 OFF, SW1-6 OFF		
Baud Rate: 50 75 110 135 150 300 600 1200 1800 2400 3600 4800 7200 9600 19200	SW1-1 to SW1-4 same as Printer Mode	POKE 1144+s,r r = (not available) 0 dec/$00 hex 176 dec/$B0 hex 144 dec/$90 hex 128 dec/$80 hex 64 dec/$40 hex 32 dec/$20 hex 16 dec/$10 hex 11 dec/$0B hex 8 dec/$08 hex 5 dec/$05 hex 4 dec/$04 hex (not available) 2 dec/$02 hex 1 dec/$01 hex	POKE 49291+s*16,r r = b + d; b = 1 dec/$01 hex 2 dec/$02 hex 3 dec/$03 hex 4 dec/$04 hex 5 dec/$05 hex 6 dec/$06 hex 7 dec/$07 hex 8 dec/$08 hex 9 dec/$09 hex 10 dec/$0A hex 11 dec/$0B hex 12 dec/$0C hex 13 dec/$0D hex 14 dec/$0E hex 15 dec/$0F hex
Data Format: 8 data,1 stop 7 data,1 stop 6 data,1 stop 5 data,1 stop 8 data,2 stop 7 data,2 stop 6 data,2 stop 5 data,2 stop	 SW2-1 ON SW2-1 OFF	POKE 1912+s,r POKE 1272+s,t r = 9; t = 1* r = 8; t = 1* r = 7; t = 1* r = 6; t = 1* r = 9; t = 2* r = 8; t = 2* r = 7; t = 2* r = 6; t = 2* * add 1 if p = 1 or 0	(to get r above, add d to b) d = 16 dec/$10 hex 48 dec/#30 hex 80 dec/$50 hex 112 dec/$70 hex 144 dec/$90 hex 176 dec/$B0 hex 208 dec/$D0 hex 240 dec/$F0 hex
Parity: none odd even MARK SPACE		POKE 1400+s,p p = 2 p = 1 p = 0 (not available) (not available)	POKE 49290+s*16,p p = 11 ($0B hex) p = 43 ($2B hex) p = 107 ($6B hex) (not available) (not available)

Table B-1. SIC Switch Settings, PEEKs and POKEs, Part I

<table>
<tr><th rowspan="2">Selection</th><th rowspan="2">SSC switches and settings</th><th colspan="2">PEEKs and POKES to use for</th></tr>
<tr><th>P8 Serial Card</th><th>Super Serial Card</th></tr>
<tr><td>Line Width:</td><td>SW2-3 & SW2-4, same as Printer Mode</td><td>POKE 1784+s,r
r=1 to 255;
for no <CR>,r=Ø</td><td>POKE 1784+s,r
r=4Ø to 255; for no <CR>, PEEK 14ØØ+s, POKE 14ØØ+s, (old value + 128)</td></tr>
<tr><td>Video/
Generate <LF>/
Translate/
<CR> Delay:</td><td>SW2-3 & SW2-4
SW2-5
(no switch)
SW2-2

(all switches same as in Printer Mode)</td><td>V = Video on?
G = Gen. <LF>?
T = LC to UC?
D = Dly 1/4 s?
POKE 2Ø4Ø+s,r
r=
dec hex V G T D
4 $Ø4 Y N Y Y
5 $Ø5 Y Y Y Y
36 $24 Y N N Y
37 $25 Y Y N Y
68 $44 Y N Y N
69 $45 Y Y Y N
1ØØ $64 Y N N N
1Ø1 $65 Y Y N N
132 $84 N N Y Y
133 $85 N Y Y Y
164 $A4 N N N Y
165 $A5 N Y N Y
196 $C4 N N Y N
197 $C5 N Y Y N
228 $E4 N N N N
229 $E5 N Y N N</td><td>V = Video on?
G = Gen. <LF>?

POKE 2Ø4Ø+s,r
r=
dec hex V G
4 $Ø4 N N
5 $Ø5 N Y
132 $84 Y N
133 $85 Y Y

T = LC to UC?
D = Dly 1/4 s?

POKE 1144+s,r
r =
dec hex T D
Ø $ØØ Y N
16 $1Ø Y Y
64 $4Ø N N
8Ø $5Ø N Y</td></tr>
</table>

Table B-2. SIC Switch Settings, PEEKs and POKEs, Part II

P8A EMULATION POKES

The P8A ROM differs from the P8 ROM in several ways:

1) The <CR> delay switch now determines whether an ETX/ACK handshake is performed after each <CR> that is transmitted. The corresponding RAM bit was not the same as the P8 <CR> delay bit, but was kept in bit 2 of location 14ØØ+s. For SSC emulation, the control is the same as the <CR> delay bit as noted above (in location 1144+s).

2) The number of stop bits was always 2; for SSC P8A mode this is configured via switch SW2-1 and can also be set via software by POKEing location 4929 as noted above.

3) The printer width information was kept in the same location that the P8 ROM kept the number of stop bits; the P8 printer width byte was zeroed to avoid automatic generation of carriage returns. The SSC P8A emulation code keeps the printer width information in the

same place as for P8 emulation and uses the high-order bit at location 14ØØ+s to control automatic generation of carriage returns.

4) Lowercase input is enabled by default for the P8A ROM; in P8A emulation, however, it is enabled by the POKE shown in Table B-2.

5) In contrast to the P8 ROM, the P8A ROM and the SSC do not support batch moves.

6) The enquire character for the SIC P8A ROM was ETX (ASCII 3); for SSC P8A mode, this can be changed to another control character by a POKE to location 14ØØ+s. For example, to change the enquire character to ENQ (ASCII 5), which is used by many RS-232 devices, use this POKE: POKE 14ØØ+s,5. Note that this also disables the automatic generation of carriage returns. Actually, any character between Ø and 31 can be used, although only 3 and 5 are used much.

OTHER EMULATION MODE DIFFERENCES

If your old programs, written to control one of the old Serial Interface Card ROMs, still don't work after you've followed all this handy advice, then read on.

The SSC always monitors the RS-232-C handshake lines to determine whether or not the device is ready to accept data. If your device fails to assert one of these lines, the SSC will wait patiently forever.

When the arrow on the jumper block is pointing toward TERMINAL, your device sees DCD and DSR asserted as soon as the SSC is initialized, and the SSC sees CTS whenever the device sends RTS. If the device does not assert both RTS and DTR, the SSC will assume it is not ready to receive data. This can be used as a hardware handshake to prevent buffer overflow at the device (e.g., when your printer runs out of paper it can stop asserting one of these lines and the SSC will wait while you put in more paper). If you do not connect these lines, the SSC will always treat them as if they were asserted.

The Serial Interface Card tied RTS to CTS, and DTR to DCD and DSR; if your RS-232 device depended upon this, you may want to make a special connector which does this.

Your device may have depended upon the half-duplex nature of the SIC. The ACIA on the SSC is able to send and receive at the same time and is always configured to do so.

The SIC was initialized each time it was called at location $CsØØ (for example, by a PR#s or IN#s). The SSC is only reintialized after the ACIA has been reset (either by resetting the Apple or by exiting from Printer or Communication Mode via a Reset command).

OLD COMMUNICATIONS CARD COMMANDS

The SSC supports all the functions supported by the old Apple II Communications Interface Card (CIC), although the two ACIAs' registers are not the same on a bit-by-bit level. The SSC also supports the CIC commands: <CTRL-T>, <CTRL-R>, and <CTRL-S>.

SWITCH TO TERMINAL MODE—<CTRL-T>

In Communication Mode, the SSC is initialized to recognize the remote-control command <CTRL-T> arriving in the stream of incoming data. This character causes the SSC to enter Terminal Mode (the same as the T(erminal command (Chapter 3). You can disable <CTRL-T> recognition by issuing an X(OFF D(isable command.

BYPASS TERMINAL MODE—<CTRL-R>

When the SSC is in Terminal Mode and X(OFF E(nable (the default in this mode) is in effect, the SSC recognizes the remote control command <CTRL-R> arriving in the input data stream, and responds by bypassing (exiting from) Terminal Mode. This is the same as the Q(uit Terminal Mode command (Chapter 3).

XOFF—<CTRL-S>

The SSC interprets <CTRL-S> as the ASCII XOFF character. When it receives <CTRL-S> from a remote device, it stops transmitting data until it receives an XON character from that device.

PARALLEL CARD COMMANDS

The SSC is not hardware compatible with the Apple II Parallel Cards. However, for the sake of compatibility with software written for parallel interface applications, the SSC supports the following commands. You do not need to follow these commands with <RETURN>.

LINE WIDTH n AND VIDEO OFF—<CTRL-I><n>N

This command turns off the Apple II video screen and generates a <CR> after n characters (if automatic <CR> generation is enabled via the C command (Chapter 2); n can be any value from 40 through 255.

LINE WIDTH 40 AND VIDEO ON—<CTRL-I>I

This command turns on the Apple II video screen and sets the line width to 40.

DISABLE AUTOMATIC LINEFEED—<CTRL-I>K

This command has the same effect as L(inefeed D(isable (Chapter 2): it turns off automatic generation of <LF> after <CR>.

APPENDIX C
SPECIFICATIONS AND SCHEMATICS

This appendix contains the SSC specifications, connector pin assignments, jumper block wiring, and a schematic diagram. Use the schematic diagram with the Theory of Operation section in Chapter 4.

SSC SPECIFICATIONS

PHYSICAL CHARACTERISTICS

Dimensions	2-3/4" x 7" (68.8 mm x 177.8 mm)
Weight	3 oz. (9Ø gm), approximately
Cables required	internal cable from 1Ø-pin header on SSC to DB-25 connector on case of Apple II (supplied); shielded RS-232-C cable to external device (not supplied)
Controls	2 blocks of 7 switches each, set by user before installation
Special Tools	none required

ENVIRONMENT

Operating temperature	4Ø° F to 95° F (5° C to 35° C)
Storage temperature	-4Ø° F to 122° F (-4Ø° C to 5Ø° C)
Operating relative humidity	5% to 95% (noncondensing)
Storage relative humidity	5% to 95% (noncondensing)

SPECIAL CIRCUITS

SY6551	Asynchronous Communications Interface Adapter
2316	Read Only Memory (2,Ø48 by 8 bits) with SSC firmware
	The SSC has the usual power supply bypassing capacitors

APPLE II SLOT LOCATION

BASIC programs	any slot except slot #Ø
APPLESOFT programs	any slot except slot #Ø
PASCAL programs	slot #1 for use with printer, etc.
	slot #2 for use with modem
	slot #3 for use with terminal

SOFTWARE COMPATIBILITY

The SSC is compatible with the following languages and operating systems:

Integer BASIC	DOS 3.2	Pascal 1.Ø	65Ø2 Assembler
Applesoft BASIC	DOS 3.3	Pascal 1.1	

Under BASIC, input sent to the SSC at high baud rates may be lost, since the SSC can only buffer two characters at a time and BASIC may not be fast enough to read characters before they are overlaid.

In any software environment, characters may be lost when sent to the video screen in scrolling mode at greater than 3ØØ baud. There are at least three solutions to this problem: lower the baud rate to 3ØØ baud; reduce the scrolling window size (using 2 fewer lines already makes 12ØØ baud possible), or use an 8Ø-column card with automatic hardware scrolling.

CONNECTOR PIN ASSIGNMENTS

Table C-1 lists the signals assigned to the connector pins on the 1Ø-pin header at location 7B on the SSC, and the corresponding pins on the DB-25 connector that you attach to the back of the Apple II case.

1Ø-pin Header	DB-25 Connector	Signal name
1	1	Frame Ground
2	2	Transmit Data (TXD)
3	3	Receive Data (RXD)
4	4	Request To Send (RTS)
5	5	Clear To Send (CTS)
6	6	Data Set Ready (DSR)
7	19	Secondary Clear To Send (SCTS)
8	7	Signal Ground
9	2Ø	Data Terminal Ready (DTR)
1Ø	8	Data Carrier Detect (DCD)

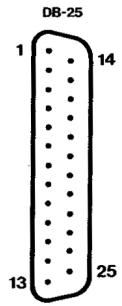


Table C-1. Connector Pin Assignments

JUMPER BLOCK WIRING

Table C-2 lists the signals that the jumper block connects to the SSC when the arrow points toward the word MODEM and when it points toward the word TERMINAL. In the latter case, the jumper block acts as a modem eliminator.

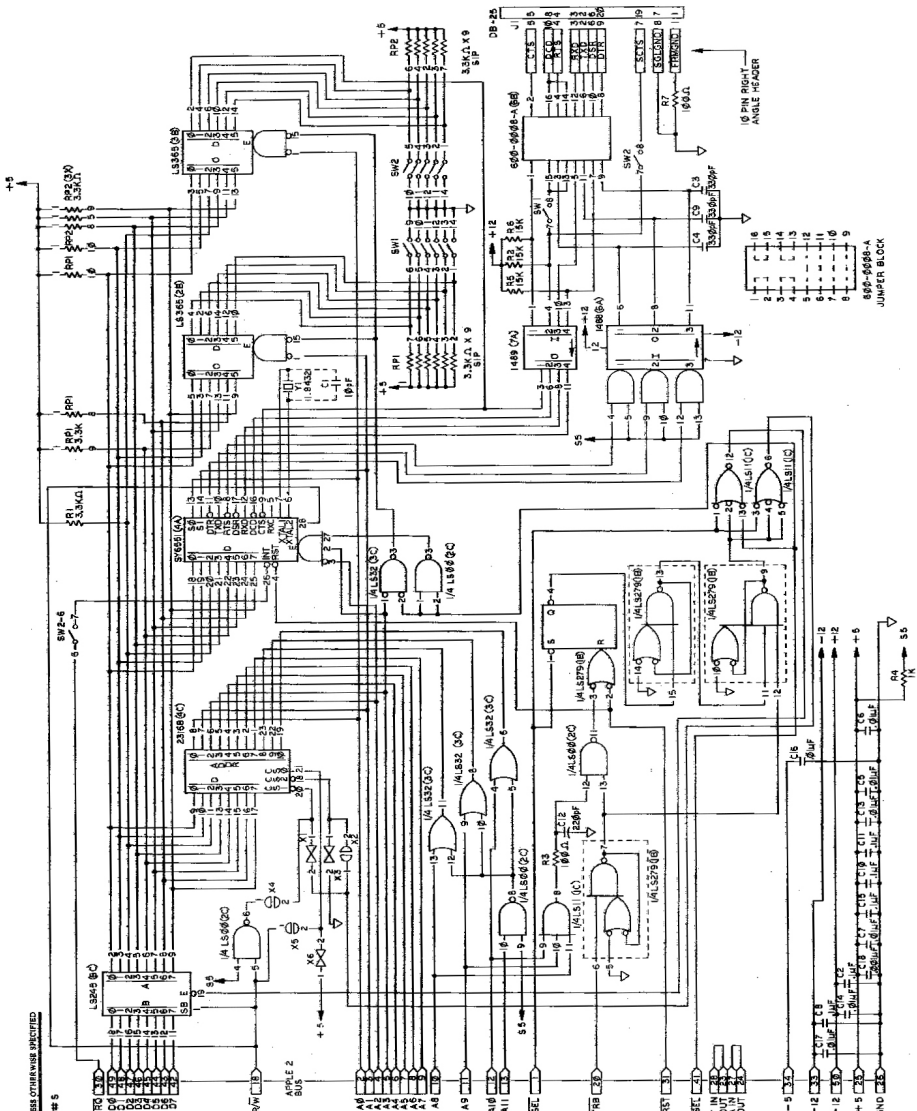
Note that all RS-232-C signals on the SSC use negative-true logic; that is, they are true (asserted) at Ø volts and false at +5 volts.

Signal at SSC	MODEM position (pin)	TERMINAL position (pin)
Transmit Data	Transmit Data (2)	Receive Data (3)
Receive Data	Receive Data (3)	Transmit Data (2)
Request To Send	Request To Send (4)	Data Carrier Detect (8)
Clear To Send	Clear To Send (5)	Data Carrier Detect (8)
Data Set Ready	Data Set Ready (6)	Data Terminal Ready (2Ø)
Data Terminal Ready	Data Term. Ready (2Ø)	Data Set Ready (6)
Data Carrier Detect	Data Carrier Detect (8)	Request To Send (4)
Data Carrier Detect	Data Carrier Detect (8)	Clear To Send (5)*

*When SW1-7 is OFF and SW2-7 is ON, the jumper block in the TERMINAL position connects Data Carrier Detect on the SSC to Secondary Clear To Send on the DB-25 connector.

Table C-2. Jumper Block Wiring

SCHEMATIC DIAGRAM



APPENDIX D
ASCII CODE TABLE

The table below shows the entire ASCII character set, and how to generate each character. Not all characters are available directly from the Apple II keyboard. However, in Terminal Mode (Chapter 3) you can generate all of the lowercase and special ASCII characters not accessible directly from the Apple II keyboard.

Here is how to interpret this table:

- The BINARY column has the 7-bit code for each ASCII character.
- The LOW DEC column gives the decimal equivalent of the 7-bit binary value. This value is the same if the binary code has 8 bits and the high-order bit is Ø (SPACE parity; Pascal).
- The LOW HEX column gives the corresponding hexadecimal value.
- The HI DEC column gives the decimal equivalent of the 7-bit binary value if a high-order bit equal to 1 is appended to it (MARK parity; BASIC); for example, 11ØØ1ØØØ for the letter H.
- The HI HEX column gives the corresponding hexadecimal value.
- The ASCII CHAR column gives the ASCII character name.
- The INTERPRETATION column spells out the meaning of special symbols and abbreviations where necessary.
- The WHAT TO TYPE column indicates what keystrokes generate the ASCII character from the NORMAL (unaided) Apple II keyboard, and from the TERMINAL Mode (firmware assisted) keyboard. Characters not accessible are labeled "n/a." The numbers between columns refer to footnotes.
- Angle brackets enclose the names of single keys (like <ESC> for the ESC key), or enclose keystrokes involving more than one key (like <CTRL-SHIFT-M>, which means "hold down CTRL and SHIFT while pressing M.") But <ESC>9 means "type ESC, THEN type 9" because the 9 is outside the angle brackets.

To put the SSC in Terminal Mode, set SW1-5 and SW1-6 both ON; then use the T command or the remote-control <CTRL-T> command. When the SSC first enters Terminal Mode, the keyboard is locked in uppercase. Press <ESC> once for lowercase. This also prepares the SSC for the special <ESC>-plus-number keystrokes. Press <ESC> twice in a row to lock the keyboard in uppercase again.

7-BIT BINARY	LOW DEC	LOW HEX	HI DEC	HI HEX	ASCII CHAR	INTERPRETATION	WHAT TO TYPE NORMAL	TERMINAL
ØØØØØØØ	Ø	ØØ	128	8Ø	NUL	Blank (null)	<CTRL-@>	
ØØØØØØ1	1	Ø1	129	81	SOH	Start of Header	<CTRL-A>	1
ØØØØØ1Ø	2	Ø2	13Ø	82	STX	Start of Text	<CTRL-B>	
ØØØØØ11	3	Ø3	131	83	ETX	End of Text	<CTRL-C>	2
ØØØØ1ØØ	4	Ø4	132	84	EOT	End of Transm.	<CTRL-D>	
ØØØØ1Ø1	5	Ø5	133	85	ENQ	Enquiry	<CTRL-E>	3
ØØØØ11Ø	6	Ø6	134	86	ACK	Acknowledge	<CTRL-F>	4
ØØØØ111	7	Ø7	135	87	BEL	Bell	<CTRL-G>	
ØØØ1ØØØ	8	Ø8	136	88	BS	Backspace	<CTRL-H>	5
ØØØ1ØØ1	9	Ø9	137	89	HT	Horizontal Tab	<CTRL-I>	6
ØØØ1Ø1Ø	1Ø	ØA	138	8A	LF	Linefeed	<CTRL-J>	
ØØØ1Ø11	11	ØB	139	8B	VT	Vertical Tab	<CTRL-K>	
ØØØ11ØØ	12	ØC	14Ø	8C	FF	Form Feed	<CTRL-L>	
ØØØ11Ø1	13	ØD	141	8D	CR	Carriage Return	<CTRL-M>	7
ØØØ111Ø	14	ØE	142	8E	SO	Shift Out	<CTRL-N>	
ØØØ1111	15	ØF	143	8F	SI	Shift In	<CTRL-O>	
ØØ1ØØØØ	16	1Ø	144	9Ø	DLE	Data Link Escape	<CTRL-P>	
ØØ1ØØØ1	17	11	145	91	DC1	Device Control 1	<CTRL-Q>	8
ØØ1ØØ1Ø	18	12	146	92	DC2	Device Control 2	<CTRL-R>	9
ØØ1ØØ11	19	13	147	93	DC3	Device Control 3	<CTRL-S>	1Ø
ØØ1Ø1ØØ	2Ø	14	148	94	DC4	Device Control 4	<CTRL-T>	11
ØØ1Ø1Ø1	21	15	149	95	NAK	Neg. Acknowledge	<CTRL-U>	12
ØØ1Ø11Ø	22	16	15Ø	96	SYN	Synchronization	<CTRL-V>	
ØØ1Ø111	23	17	151	97	ETB	End of Text Blk.	<CTRL-W>	
ØØ11ØØØ	24	18	152	98	CAN	Cancel	<CTRL-X>	
ØØ11ØØ1	25	19	153	99	EM	End of Medium	<CTRL-Y>	
ØØ11Ø1Ø	26	1A	154	9A	SUB	Substitute	<CTRL-Z>	
ØØ11Ø11	27	1B	155	9B	ESC	Escape	<ESC>	13 <ESC>Ø

1. Normal command character in Communication Mode.
2. Used in ETX/ACK protocol (SIC P8A Emulation Mode).
3. Used in ENQ/ACK protocol (SIC P8A Emulation Mode).
4. Used in ETX/ACK or ENQ/ACK protocol (SIC P8A Emulation Mode).
5. Or use ← key.
6. Normal Command character in Printer Mode.
7. Or use <RETURN> key.
8. XON in XON/XOFF protocol (usually in Communication Mode).
9. Remote-control command to Exit from Terminal Mode.
1Ø. XOFF in XON/XOFF protocol (usually in Communication Mode).
11. Remote-control command to Enter Terminal Mode.
12. Or use → key.
13. Use the ESC key to generate the Escape character with the normal Apple II keyboard. In Terminal Mode, use <ESC>Ø.

7-BIT BINARY	LOW DEC	LOW HEX	HI DEC	HI HEX	ASCII CHAR	INTERPRETATION	WHAT TO TYPE NORMAL	TERMINAL
ØØ111ØØ	28	1C	156	9C	FS	File Separator	n/a	<ESC>1
ØØ111Ø1	29	1D	157	9D	GS	Group Separator	<CTRL-SHIFT-M>	
ØØ1111Ø	3Ø	1E	158	9E	RS	Record Separator	<CTRL-SHIFT-N>	
ØØ11111	31	1F	159	9F	US	Unit Separator	n/a	<ESC>2
Ø1ØØØØØ	32	2Ø	16Ø	AØ	SP	Space	spacebar	
Ø1ØØØØ1	33	21	161	A1	!		!	
Ø1ØØØ1Ø	34	22	162	A2	"		"	
Ø1ØØØ11	35	23	163	A3	#		#	
Ø1ØØ1ØØ	36	24	164	A4	$		$	
Ø1ØØ1Ø1	37	25	165	A5	%		%	
Ø1ØØ11Ø	38	26	166	A6	&		&	
Ø1ØØ111	39	27	167	A7	'	Closing Quote	'	
Ø1Ø1ØØØ	4Ø	28	168	A8	(		(	
Ø1Ø1ØØ1	41	29	169	A9	)		)	
Ø1Ø1Ø1Ø	42	2A	17Ø	AA	*		*	
Ø1Ø1Ø11	43	2B	171	AB	+		+	
Ø1Ø11ØØ	44	2C	172	AC	,	Comma	,	
Ø1Ø11Ø1	45	2D	173	AD	-	Hyphen	-	
Ø1Ø111Ø	46	2E	174	AE	.	Period	.	
Ø1Ø1111	47	2F	175	AF	/		/	
Ø11ØØØØ	48	3Ø	176	BØ	Ø		Ø	
Ø11ØØØ1	49	31	177	B1	1		1	
Ø11ØØ1Ø	5Ø	32	178	B2	2		2	
Ø11ØØ11	51	33	179	B3	3		3	
Ø11Ø1ØØ	52	34	18Ø	B4	4		4	
Ø11Ø1Ø1	53	35	181	B5	5		5	
Ø11Ø11Ø	54	36	182	B6	6		6	
Ø11Ø111	55	37	183	B7	7		7	
Ø111ØØØ	56	38	184	B8	8		8	
Ø111ØØ1	57	39	185	B9	9		9	
Ø111Ø1Ø	58	3A	186	BA	:		:	
Ø111Ø11	59	3B	187	BB	;		;	
Ø1111ØØ	6Ø	3C	188	BC	<		<	
Ø1111Ø1	61	3D	189	BD	=		=	
Ø1111Ø	62	3E	19Ø	BE	>		>	
Ø111111	63	3F	191	BF	?		?	
1ØØØØØØ	64	4Ø	192	CØ	@		@	
1ØØØØØ1	65	41	193	C1	A		A	
1ØØØØ1Ø	66	42	194	C2	B		B	
1ØØØØ11	67	43	195	C3	C		C	
1ØØØ1ØØ	68	44	196	C4	D		D	
1ØØØ1Ø1	69	45	197	C5	E		E	
1ØØØ11Ø	7Ø	46	198	C6	F		F	
1ØØØ111	71	47	199	C7	G		G	
1ØØ1ØØØ	72	48	2ØØ	C8	H		H	
1ØØ1ØØ1	73	49	2Ø1	C9	I		I	
1ØØ1Ø1Ø	74	4A	2Ø2	CA	J		J	
1ØØ1Ø11	75	4B	2Ø3	CB	K		K	
1ØØ11ØØ	76	4C	2Ø4	CC	L		L	
1ØØ11Ø1	77	4D	2Ø5	CD	M		M	
1ØØ111Ø	78	4E	2Ø6	CE	N		N	

7-BIT BINARY	LOW DEC	LOW HEX	HI DEC	HI HEX	ASCII CHAR	INTERPRETATION	WHAT TO TYPE NORMAL		TERMINAL
1ØØ1111	79	4F	2Ø7	CF	O		O		
1Ø1ØØØØ	8Ø	5Ø	2Ø8	DØ	P		P		
1Ø1ØØØ1	81	51	2Ø9	D1	Q		Q		
1Ø1ØØ1Ø	82	52	21Ø	D2	R		R		
1Ø1ØØ11	83	53	211	D3	S		S		
1Ø1Ø1ØØ	84	54	212	D4	T		T		
1Ø1Ø1Ø1	85	55	213	D5	U		U		
1Ø1Ø11Ø	86	56	214	D6	V		V		
1Ø1Ø111	87	57	215	D7	W		W		
1Ø11ØØØ	88	58	216	D8	X		X		
1Ø11ØØ1	89	59	217	D9	Y		Y		
1Ø11Ø1Ø	9Ø	5A	218	DA	Z		Z		
1Ø11Ø11	91	5B	219	DB	[	Opening Bracket	n/a		<ESC>3
1Ø111ØØ	92	5C	22Ø	DC	\	Reverse Slant	n/a		<ESC>4
1Ø111Ø1	93	5D	221	DD	]	Closing Bracket	<SHIFT-M>		
1Ø1111Ø	94	5E	222	DE	^	Circumflex	^		
1Ø11111	95	5F	223	DF	_	Underline	n/a		<ESC>5
11ØØØØØ	96	6Ø	224	EØ	`	Opening Quote	n/a	15	
11ØØØØ1	97	61	225	E1	a		n/a		a
11ØØØ1Ø	98	62	226	E2	b		n/a		b
11ØØØ11	99	63	227	E3	c		n/a		c
11ØØ1ØØ	1ØØ	64	228	E4	d		n/a		d
11ØØ1Ø1	1Ø1	65	229	E5	e		n/a		e
11ØØ11Ø	1Ø2	66	23Ø	E6	f		n/a		f
11ØØ111	1Ø3	67	231	E7	g		n/a		g
11Ø1ØØØ	1Ø4	68	232	E8	h		n/a		h
11Ø1ØØ1	1Ø5	69	233	E9	i		n/a		i
11Ø1Ø1Ø	1Ø6	6A	234	EA	j		n/a		j
11Ø1Ø11	1Ø7	6B	235	EB	k		n/a		k
11Ø11ØØ	1Ø8	6C	236	EC	l		n/a		l
11Ø11Ø1	1Ø9	6D	237	ED	m		n/a		m
11Ø111Ø	11Ø	6E	238	EE	n		n/a		n
11Ø1111	111	6F	239	EF	o		n/a		o
111ØØØØ	112	7Ø	24Ø	FØ	p		n/a		p
111ØØØ1	113	71	241	F1	q		n/a		q
111ØØ1Ø	114	72	242	F2	r		n/a		r
111ØØ11	115	73	243	F3	s		n/a		s
111Ø1ØØ	116	74	244	F4	t		n/a		t
111Ø1Ø1	117	75	245	F5	u		n/a		u
111Ø11Ø	118	76	246	F6	v		n/a		v
111Ø111	119	77	247	F7	w		n/a		w
1111ØØØ	12Ø	78	248	F8	x		n/a		x
1111ØØ1	121	79	249	F9	y		n/a		y
1111Ø1Ø	122	7A	25Ø	FA	z		n/a		z
1111Ø11	123	7B	251	FB	{	Opening Brace	n/a		<ESC>6
1111ØØ	124	7C	252	FC	|	Vertical Line	n/a		<ESC>7
1111Ø1	125	7D	253	FD	}	Closing Brace	n/a		<ESC>8
1111Ø1Ø	126	7E	254	FE	~	Overline (Tilde)	n/a		<ESC>9
1111111	127	7F	255	FF	DEL	Delete/Rubout	n/a		<ESC>:

15. Use Closing Quote (39). For high value, use CHR$(96), etc.

APPENDIX E
TROUBLESHOOTING HINTS

This appendix contains two tables designed to help you diagnose problems that can occur when using the SSC to communicate with an RS-232-C device. The device can be a printer, or a plotter, or terminal, or another computer, or some other Data Terminal Equipment (DTE), and it can be connected either directly, or via a modem or some other Data Communication Equipment (DCE). Whenever two DTEs are connected together, there must be TWO modems (DCEs) or ONE modem eliminator (such as the jumper block when it points toward the word TERMINAL) between them.

When diagnosing problems, remember that there are many variables involved in the communications connection:

- the Apple II and its keyboard, screen, and software
- the SSC, the slot it is in, its switch settings (especially mode selection), its jumper block, cable, and software commands
- the external cable, with some number of wires (enough wires?) connected to pins (all the correct pins?) at each end
- possibly two modems connected by low-grade telephone lines, plus another cable from the remote modem to the remote device
- an RS-232-C device at the other end, with its own switch settings and needs (such as paper, ribbon, AC power...)

As you can see, making all these components work together correctly is no mean feat. If there are problems, the easiest way to resolve them is to start with very simple, sure communication between the Apple and the device. Once you have established basic communication (even if the characters are garbled), further troubleshooting becomes much easier. Be patient and methodical.

Trouble usually has characteristics visible on the Apple II screen (Table E-1), or at the device (Table E-2). If your troubleshooting efforts fail, consult your Apple dealer--but first record all the variables (as outlined above) and the symptoms you observed.

Problem	Symptom	Possible Cause	Solution
no data transfer	no sign of any communication at all	cable wires not connected OK; jumper block facing wrong way	check all cable connections, then pin assignments; try reversing jumper block
characters garbled	jh2 3g%$Q	wrong baud rate	change SW1-1 TO SW1-4 or use <n>B command
		wrong data format	change SW2-1 (and SW2-2 in Comm Mode) or use <n>D command to change format
		other device is off, out of paper, etc., off-line	turn on device, remedy its problems, put it on-line
paper not advancing	one line of smudge	printer needs line feeds from SSC	turn SW2-5 ON or use L(inefeed E(nable command
printer is skipping lines	lines look like this	printer and SSC both generating <LF> after <CR>	turn off SW2-5 in Printer Mode, or use L(inefeed D(isable command
missing characters	mssig caractrs	device buffer is overflowing	if device supports full RS-232-C handshaking, ensure all required cable wires are connected if device supports only ETX/ACK, set SIC P8A Mode if device supports XON/XOFF, set Printer Mode and use X(OFF E(nable cmd or set Comm Mode if device supports none of these, set delays with <n>C, <n>L and <n>F cmds
device sticks at line's end going nuts	one long OK line, smudge at right end	device doesn't generate own <CR>, and isn't getting enough from Apple	use SIC P8 Mode and <n>N command, or Printer Mode and C command plus appropriate SW2-3 and SW2-4 have software send <CR> before right margin

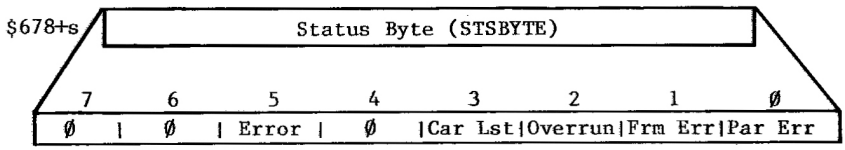
Table E-1. Problems Detected at the Device

Problem	Symptom	Possible Cause	Solution
Apple has occasional bad times	it works one minute & not next	ACIA interrupting the Apple when DCD or DSR changes	make sure that interrupt switch SW2-6 is OFF
Apple not working	dead kybd and screen	SSC in slot #3 under Pascal	Pascal expects external terminal to run the show
Apple kybd seems off	keystrokes all lost	echo off; keyboard zapped; IN# not Ø	use E(cho E(nable cmd; unzap with POKE; IN#Ø
screen seems off	nothing typed is displayed	device not echoing (half duplex) or ACIA not sending to screen	in Comm or Terminal Mode, use E(cho E(nable; in SIC or Printer Mode, use I command or SW2-3 & -4 ON
screen is seeing double	eevveerryy tthhiinngg ttwwiiccee	device & SSC both echoing to Apple (full duplex)	use E(cho D(isable cmd in Comm Mode or use <n>N cmd in Printer Mode
screen is spacing double	lines look like this	device generating and sending <LF> after <CR>	use M(ask E(nable command to remove extra linefeeds
forced uppercase display	lowercase beCOMES UPPERCASE	Apple monitor changing letters in GETLINE routine	use <n>T command to allow lowercase to pass through (not possible in Pascal)
Apple misses some characters at the beginning of lines	pple sses ome racters t the bgnning lines	screen scrolling too slowly, or BASIC or Pascal program running too slowly, and so ACIA overruns	turn off screen (<n>N or SW2-3 & -4 in Prtr Mode); reduce scroll window; use assembly language or faster program routines; use lower baud rate (3ØØ vs. 12ØØ); use <n>C, <n>L or <n>F commands; in Comm Mode, chain (<n>S cmd) to 8Ø-column card with its own scrolling hardware

Table E-2. Problems Detected at the Apple

APPENDIX F
ERROR CODES

The SSC uses I/O scratchpad address $678+s (s is the number of the slot that the SSC is in) to record status after a read operation. The firmware calls this byte STSBYTE. Table F-1 lists the bit definitions of this byte:



Bit	"1" Means	"Ø" Means
Ø	Parity Error occurred	No Parity Error occurred
1	Framing Error occurred	No Framing Error occurred
2	Overrun occurred	No Overrun occurred
3	Carrier lost	Carrier present
5	Error occurred	No error occurred

Table F-1. STSBYTE Bit Definitions

The terms Parity Error, Framing Error and Overrun are defined in the Glossary.

Bits Ø,1, and 2 are the same as the corresponding three bits of the ACIA Status Register (Appendix A). Bit 3 indicates whether or not the Data Carrier Detect (DCD; Chapter 4) signal went false at any time during the receive operation. Bit 5 is set if any of the other bits are set, as an overall error indicator. If bit 5 is the only bit set, an unrecognized command was detected. If all bits are Ø, no error occurred.

In BASIC, you can check this status byte via a PEEK $678+s (s is the SSC slot), and reset it with a POKE command at the same location.

In Pascal, the IORESULT function returns the error code value.

Any character--including the carriage return at the end of a WRITELN statement--will cause posting of a new value in IORESULT.

Table F-2 shows the possible combinations of error bits correspond to these decimal error codes.

BASIC PEEK $678+s or Pascal IORESULT	Carrier Lost	Overrun	Framing Error	Parity Error
Ø		(no error)		
32		(illegal command)		
33	no	no	no	yes
34	no	no	yes	no
35	no	no	yes	yes
36	no	yes	no	no
37	no	yes	no	yes
38	no	yes	yes	no
39	no	yes	yes	yes
4Ø	yes	no	no	no
41	yes	no	no	yes
42	yes	no	yes	no
43	yes	no	yes	yes
44	yes	yes	no	no
45	yes	yes	no	yes
46	yes	yes	yes	no
47	yes	yes	yes	yes

Table F-2. Error Codes and Bits

These error codes begin with the number 32 to avoid conflicting with previously defined and documented system error codes.

GLOSSARY

To avoid lengthy or repetitive definitions, many terms used in one definition are themselves defined elsewhere in this glossary. Also for the sake of brevity, terms and expressions are spelled out, with their abbreviations immediately after them. In a glossary of this size, the reader will have little difficulty locating abbreviations.

ACK: An ASCII character (decimal 6; Appendix D) sent from a device to the Apple II in response to an ETX or ENQ character in SIC P8A Emulation Mode.

American Standard Code for Information Interchange (ASCII): A standard defining the codes to represent a 128-element character set (Appendix D) in a fixed way for devices of different manufacturers. It is the standard for digital communication over telephone lines.

Asserted: Made true (positive in positive-true logic; negative in negative-true logic). Usually refers to electrical signals, like the RS-232-C signal Clear To Send, etc.

Asynchronous: Having a variable time interval between characters.

Asynchronous Communications Interface Adapter (ACIA): In the SSC, a single chip (Synertek 6551 or equivalent) that converts data from parallel to serial form and vice versa, and handles serial transmission and reception and RS-232-C signals, under the control of internal registers set and changed by SSC firmware.

Baud: A unit of signalling speed equal to the number of discrete conditions or signal events per second. With the SSC, for example, using a data format of 1 start bit, 7 data bits, 1 parity bit and 1 stop bit (1Ø bits in all), 3ØØ baud is approximately equal to 3Ø characters per second.

Binary: A number system with two digits, "Ø" and "1," with each digit position moving from right to left representing a successive power of two. For example, 1 represents decimal 1; 1Ø represents 2; 1ØØ represents 4; 1ØØØ represents 8, etc.

Bit: A BInary digiT, either a Ø or a 1.

BREAK: A Ø.233 second SPACE (Ø) signal sent over a communication line to interrupt the sender. This signal is often used to end a session with a timesharing service.

Carriage Return (CR): An ASCII character (decimal 13; Appendix D) that ordinarily causes a printer or display screen to place the subsequent character on the left margin. On a manual typewriter, this movement is combined with linefeed (the advancement of the paper to the next line). With computers, carriage return and linefeed are separate, causing hair-raising problems for the user.

Carrier: The background signal on a communication channel that is modified to "carry" the information. Under RS-232-C, the carrier signal is equivalent to a continuous MARK or 1; a transition to Ø then represents a start bit.

Character: Any symbol that has a widely understood meaning. In the ASCII code, letters, numbers, punctuation marks, and so on, are all characters (Appendix D).

Chip: A tiny wafer of silicon, with conductive metallic impurities, that has layers of microscopic circuits etched on it.

Clear To Send (CTS): An RS-232-C signal from a DCE to a DTE that the SSC keeps false until the DCE makes it true, indicating that all circuits are ready to transfer data.

Command Character: An ASCII character, usually <CTRL-A> or <CTRL-I> (Appendix D), that causes the SSC firmware to interpret subsequent characters as a command.

Command Register: An ACIA location (at hexadecimal address $CØ8A+sØ) that stores parity type and RS-232-C signal characteristics.

Communications Interface Card (CIC): An Apple II interface card designed to connect the Apple II to a device via a DCE.

Communications Mode: An operating state in which the SSC is prepared to exchange data and signals with a DCE.

Control Character: Any character generated by holding down the key marked CTRL while pressing some other key.

Control Register: An ACIA location (at hexadecimal address $CØ8B+sØ) that stores data format and baud rate selections.

Daisy Chaining: A method of passing incoming signals and data from one peripheral connector slot to another, such as from the SSC slot to a slot containing an 8Ø-column-display card.

Data Bit: With the SSC, one of 5 to 8 bits representing a character.

Data Carrier Detect (DCD): An RS-232-C signal from a DCE to a DTE (such as the Apple II) indicating that a communication connection has been established. The SSC's internal circuits hold DCD false until the external device sets DCD true.

Data Communication Equipment (DCE): As defined by the RS-232-C standard, any device that transmits or receives information. Usually this is a modem. However, when a Modem Eliminator is used, the Apple II looks like a DCE to the other device, and the other device looks like a DCE to the Apple.

Data Conversion: Changing of data from parallel to serial form or from serial to parallel form.

Data Format: The form in which data is stored, manipulated or transferred. Serial data transmitted and received by the SSC has a data format of: one start bit, 5 to 8 data bits, an optional parity bit, and one, one and a half, or two stop bits.

Data Set Ready (DSR): An RS-232-C signal from a DCE to a DTE indicating that the DCE has established a connection.

Data Terminal Equipment (DTE): As defined by the RS-232-C standard, any device that generates or absorbs information, thus acting as a terminus of a communication connection.

Data Terminal Ready (DTR): An RS-232-C signal from a DTE to a DCE indicating a readiness to transmit or receive data.

Default Value: A value that is assumed or set in the absence of explicit instructions otherwise.

Device: A piece of equipment; usually a printer, plotter, terminal or computer. When the jumper block is in the MODEM position, the SSC expects the device to be a DCE (such as a modem).

Echo: To send an input character to a video screen, printer, or other output device. On a typewriter, what we strike on the keyboard appears on the page in the same step. With a computer, these two steps are controlled separately.

Electromagnetic Interference (EMI): Electrical or magnetic signals or noise that disturbs the operation of radio or television receivers. For example, a hair dryer often creates EMI that fuzzes up the picture on a nearby television set.

Emulation Mode: A manner of operating in which one computer or interface imitates another. For example, in SIC P8 Emulation Mode, the SSC acts very much like an Apple II Serial Interface Card with the P8 version of firmware.

ENQ: An ASCII character (decimal 5; Appendix D) used in the ENQ/ACK protocol (SIC P8A Emulation Mode).

ETX: An ASCII character (decimal 3; Appendix D) used in the ETX/ACK protocol (SIC P8A Emulation Mode).

Even Parity: Use of an extra bit set to Ø or 1 as necessary to make the total number of 1 bits an even number. For example, the 7-bit ASCII code for the letter A (1ØØØØØ1) has two 1 bits; for even parity, the transmitting device appends an eighth bit equal to Ø so that the total number of 1 bits remains even. The receiving device can count 1 bits as a way of checking for transmission errors.

False: Zero or negative voltage in positive-true logic; positive voltage in negative-true logic. Absence of an arbitrary signal or condition.

Firmware (FW): Software that resides in ROM and so is relatively unchangeable (firm) compared to software in RAM.

Form Feed (FF): An ASCII character (decimal 12; Appendix D) that causes a printer or other paper-handling device to advance to the top of the next page.

Framing Error (FRM): Absence of the expected stop bit(s) on a received character. The ACIA records this error by setting bit 1 (FRM) of its Status Register to 1. The ACIA checks and records each framing error separately: if the next character is OK, the FRM bit is cleared.

Full Duplex: Capable of simultaneous two-way communications.

Half Duplex: Capable of communications in one direction at a time.

Handshake : A kind of communication protocol in which the receiving device, when it has successfully gotten a character or block of characters, sends back an acknowledging signal, thereby triggering the next transmission.

Hardware: The actual physical switches, wires, chips, PC boards, and so on, of a computer system.

Header: A cable connector mounted on a PC board.

Hexadecimal: A numbering system that uses 16 digits; usually these are represented by the ten decimal digits, Ø through 9, plus the letters A through F (A representing decimal ten, F representing decimal fifteen, etc.). Each hexadecimal digit can represent a string of four binary digits.

High-order Bit: See Most Significant Bit.

Initialization: The process of setting up initial values and conditions. In the SSC, the firmware finds out the switch positions and the current operating system, and uses these

findings to initialize both the ACIA registers and the Scratchpad RAM locations for the slot the SSC is in.

Input: Data that flows from the outside world into the Apple II.

Interface: Some combination of hardware, firmware and software that makes possible the useful connection of two otherwise incompatible pieces of equipment.

Interrupt: A special control signal from an external source that diverts the Apple II from the program it is executing to a specific routine that handles the condition (such as a printer gone awry) that caused the interrupt.

Jumper Block: In the SSC, a plastic plug with pins connected in such a way that it passes RS-232-C signals between the SSC and the external device either unchanged (MODEM position) or permuted in the manner of a Modem Eliminator (TERMINAL position).

Least Significant Bit (LSB): The right-hand bit of a binary number as written down; its positional value is Ø or 1 (that is, Ø or 1 times 2 to the Ø power).

Linefeed (LF): An ASCII character (decimal 1Ø; Appendix D) that ordinarily causes a printer or video display to advance to the next line.

Local: Nearby; capable of direct connection using wires only.

Low-order Bit: See Least Significant Bit.

MARK Parity: A bit of value 1 appended to the high-order end of a binary number for transmission. The receiving device can then check for errors by looking for this value on each character.

Mode: Manner of operating. The SSC can operate in one of four chief modes, depending on the settings of switches SW1-5 and SW1-6: Printer Mode, Communications Mode, SIC P8 Emulation Mode, and SIC P8A Emulation Mode.

Modem: MOdulator/DEModulator; a DCE device that connects a DTE to communications lines. As used with the SSC, a device that exchanges RS-232-C signals with the ACIA to establish a communications connection, and then either converts data from RS-232-C voltages to RS-232-C tones for transmission, or performs the opposite conversion on received data.

Modem Eliminator: The physical crossing of wires that replaces a pair of modems for direct connection of two pieces of RS-232-C Data Terminal Equipment. In the SSC, the jumper block serves this purpose when installed in the TERMINAL position.

Most Significant Bit (MSB): The leftmost bit of a binary number as written down. This bit represents Ø or 1 times 2 to the power one less than the total number of bits in the binary number. For example, in the binary number 1ØØØØ, the 1 represents 1 times 2 to the fourth power, or sixteen.

Odd Parity: Use of an extra bit set to Ø or 1 as necessary to make the total number of 1 bits an odd number. For example, the 7-bit ASCII code for the letter A (1ØØØØØ1) has two 1 bits; for odd parity, the transmitting device appends an eighth bit equal to 1, making the total number of 1 bits odd. The receiving device can check for transmission errors by counting 1 bits.

Output: Data that flows from the Apple II to an external device.

Overrun (OVR): A condition that occurs when the Apple II processor does not retrieve a received character from the Receive Data Register before the subsequent character arrives. The ACIA automatically sets bit 2 (OVR) of its Status Register; subsequent characters are lost. The Receive Data Register contains the last valid data word received.

P8: One of two types of Programmable ROM (PROM) installed in the Apple II Serial Interface Card. This PROM performed batch moves, but had no provision for software handshaking.

P8A: One of two types of Programmable ROM (PROM) installed in the Apple II Serial Interface Card. This PROM provided the ENQ/ACK software handshaking required by several types of printers.

Parallel Interface: A connection between two devices where there is a separate wire for each bit of a character, so that an entire character can be transferred in a single instant.

Parity: Maintenance of a sameness of level or count, usually the count of 1 bits in each character, for error checking. In the SSC, the ACIA has a register that stores the type of parity selected (none, odd, even, MARK or SPACE). It automatically generates the parity bit when transmitting, and both checks and discards parity bits appended to received characters.

Parity Error (PAR): Absence of the correct parity bit value in a received character. The ACIA records this error by setting bit Ø (PAR) of its Status Register to 1.

Peripheral Connector Slot: One of eight 5Ø-pin slots inside the Apple II case near the back. Within certain restrictions, each slot can contain add-on memory, an adapter for 8Ø-column display, or an interface to an external device.

Polarized Header: On the SSC, a 1Ø-pin female connector for the internal cable; this connector has a slot on one side that receives a "key" on the cable's male connector.

Printed Circuit (PC) Board: A sheet of stiff nonconductive material with one or more thin layers of metal bonded to it. Unwanted areas of this metal are etched away, leaving the paths of the desired circuits. Electronic components can then be soldered to the board. Small PC boards are also called cards.

Printer Mode: An operating state in which the SSC is prepared to exchange data and signals with another DTE (such as a printer).

Protocol: A predefined exchange of control signals between devices enabling them to prepare for coordinated data transfer.

Radio Frequency Interference (RFI): Electromagnetic interference occurring at frequencies used for radio communications.

Random Access Memory (RAM): A series of storage locations that can be accessed directly (by means of horizontal and vertical coordinates) for both reading and writing.

Read Only Memory (ROM): A series of storage locations that can be read but cannot be written to; this protects the programs and data in the ROM from alteration or destruction.

Receive Data Register: A read-only register in the ACIA (at hexadecimal location $C0\emptyset8+s\emptyset$) that stores the most recent character successfully received.

Remote: Too distant for direct connection via wires or cables only.

Request To Send (RTS): An RS-232-C signal from a DTE to a DCE to prepare the DCE for data transmission.

Ring Indicator (RI): An optional RS-232-C signal from a DCE to a DTE that indicates the arrival of a call.

RS-232-C: A standard created by the Electronic Industries Association (EIA) to allow devices of different manufacturers to exchange serial data--particularly via telephone lines. The ACIA in the SSC implements all the required primary RS-232-C signals. These signals are true when at $\emptyset$ volts.

Scratchpad RAM: Eight locations in the Apple's memory reserved for each of the 8 peripheral connector slots (64 bytes in all).

Secondary Clear To Send (SCTS): A secondary RS-232-C signal that some printers use instead of Clear To Send.

Serial Interface: A connection in which all the bits of a character are sent along a single wire one after the other.

Serial Interface Card (SIC): An Apple II product designed to connect an RS-232-C device directly to the Apple II.

SIC Emulation Mode: A state of operation in which the SSC imitates an Apple II Serial Interface Card.

SPACE Parity: A bit of value Ø appended to a binary number for transmission. The receiving device can look for this value on each character as a means of error checking.

Start Bit: A transition from a MARK signal to a SPACE signal for one bit-time, indicating that the next string of bits represents a character.

Status Register: An ACIA register (hexadecimal location $CØ89+sØ) that stores the state of two of the RS-232-C signals and of the Transmit and Receive Data Registers, as well as the outcome of the most recent character transfer.

Stop Bit: A MARK signal following a string of data bits to indicate the end of a character.

Super Serial Card (SSC): The interface card described in this manual. It is called "super" because it can simultaneously transmit and receive data in one of 35 formats at any of 15 speeds, honor several software protocols, communicate directly with either DTE or DCE, change operating characteristics in response to software commands, and dovetail with the chief operating environments offered with the Apple II.

Terminal: An input/output device, usually made up of a keyboard and video display and sometimes including its own printer and magnetic storage devices, that can act as a separate and even remote site for data transfer with a computer system.

Terminal Mode: An operating state of the SSC in which the firmware bypasses the Apple II's central processor, and makes the Apple act as a simple terminal capable of generating all of the ASCII characters.

Transmit Data Register: A write-only register in the ACIA (at hexadecimal location $CØ88+sØ) that holds the current character to be transmitted.

True: Positive voltage in positive-true logic; zero or negative voltage in negative-true logic. Assertion of an arbitrary signal or condition.

XOFF: An ASCII character (decimal 19; Appendix D) sent by a receiving device to a transmitting device to halt transmission of characters.

XON: An ASCII character (decimal 17; Appendix D) used in the XON/XOFF protocol as a go-ahead character from the receiving device to the sending device after an XOFF has been sent to halt transmission.

FIGURES AND TABLES

Chapter 1
GETTING STARTED

1 Figure 1-1 Photo of the Super Serial Card
2 Figure 1-2 Line Drawing of the SSC
3 Figure 1-3 Components of the Internal Cable and Clamp Assembly
4 Figure 1-4 Sliding the "Key" into the Groove
4 Figure 1-5 Internal Cable Attached Correctly to SSC

Chapter 2
PRINTER MODE

5 Figure 2-1 SSC Set for Printer Mode
10 Figure 2-2 SSC in Slot #1 and Clamp Assembly in Notch

6 Table 2-1 Commonly Used Switch Settings for Printer Mode
7 Table 2-2 Baud Rate Switch Settings
8 Table 2-3 Line Width and Video Switch Settings
14 Table 2-4 Printer Mode Commands
15 Table 2-5 Baud Rate Selections
15 Table 2-6 Data Format Selections
16 Table 2-7 Parity Selections
16 Table 2-8 Time Delay Selections
18 Table 2-9 Lowercase Character Displays

Chapter 3
COMMUNICATIONS MODE

21 Figure 3-1 SSC Set for Communications Mode
26 Figure 3-2 SSC in Slot #2 and Clamp Assembly in Notch

22 Table 3-1 Commonly Used Switch Settings for Communications Mode
23 Table 3-2 Baud Rate Switch Settings
23 Table 3-3 Data Format Selections
29 Table 3-4 Summary of Communications Mode Commands
30 Table 3-5 Baud Rate Selections
30 Table 3-6 Data Format Selections
31 Table 3-7 Parity Selections
32 Table 3-8 Time Delay Selections
33 Table 3-9 Lowercase Character Displays
35 Table 3-10 Special ASCII Character Generation
35 Table 3-11 Terminal Mode Commands

Chapter 4
HOW THE SSC WORKS

39 Figure 4-1 Parallel Data Transfer
39 Figure 4-2 Serial Data Transfer
39 Figure 4-3 Parallel-to-Serial Data Conversion
40 Figure 4-4 RS-232-C Serial Data Format
42 Figure 4-5 An RS-232-C Setup with Modems
43 Figure 4-6 An RS-232-C Setup with a Modem Eliminator
43 Figure 4-7 SSC Operating Modes
44 Figure 4-8 SSC Configurations
45 Figure 4-9 Overall Block Diagram of the SSC

38 Table 4-1 Binary and Decimal Digits and Quantities
41 Table 4-2 RS-232-C Signals as Interpreted by the Sender
41 Table 4-3 RS-232-C Signals as Interpreted by the Receiver
46 Table 4-4 SSC Address Remapping
47 Table 4-5 Registers in SSC Peripheral I/O Space

Appendix A
FIRMWARE

50 Table A-1 I/O Routine Offsets and Registers under Pascal 1.1
51 Table A-2 Bytes Used for Device Identification
51 Table A-3 Device Class Digit
52 Table A-4 Memory Usage Map
52 Table A-5 Zero-Page Locations Used by SSC
53 Table A-6 Scratchpad RAM Locations Used by SSC
54 Table A-7 SSC Registers in Peripheral Card I/O Space
55 Table A-8 Monitor ROM Entry Points Used by SSC
55 Table A-9 BASIC Entry Points Used by SSC
56 Table A-10 Pascal 1.0 Entry Points Used by SSC
57 Table A-11 Pascal 1.1 Offsets Used by SSC
57 Table A-12 SSC Special Firmware Locations

Appendix B
APPLE INTERFACE CARD EMULATION

93 Table B-1 SIC Switch Settings, PEEKs and POKEs, Part I
94 Table B-2 SIC Switch Settings, PEEKs and POKEs, Part II

Appendix C
SPECIFICATIONS AND SCHEMATICS

98 Table C-1 Connector Pin Assignments
99 Table C-2 Jumper Block Wiring

Appendix E
TROUBLESHOOTING HINTS

106 Table E-1 Problems Detected at the Device
107 Table E-2 Problems Detected at the Apple

Appendix F
ERROR CODES

109 Table F-1 STSBYTE Bit Definitions
110 Table F-2 Error Codes and Bits

INDEX

A

A register 50
ACIA 45, 47, 48, 91, 95, 96, 97
  function 47
  registers 49, 54
  status register 109
ACK 91, 94, 102, 106
address
  bus 45
  lines 46
  mapping 46
  space 46
addressing logic 45, 46
answer modem 42
Apple II Communications Card (see CIC)
Apple II Communications Interface Card (see CIC)
Apple II Parallel Cards 96
Apple II Serial Interface Card (see SIC)
APPLE SSC: 8, 12, 27
Applesoft 36, 98
ASCII 24, 34, 38, 95, 96
  code table 101-104
  special characters 35, 36, 101
assembly language 11, 27, 98
Asynchronous Communications Interface Adapter (see ACIA)

B

B command 35, 36
BASIC 8, 11, 12, 17, 18, 19, 27, 31, 33, 36, 92, 101, 104
  entry points 55
  error codes 109, 110
  protocol 50
batch moves 95
baud rate 6, 15, 22, 30
  selection 15, 30
  switch settings 7, 23, 47, 91, 92, 93, 98
binary 37, 101
  data 24, 38
binary/decimal representation 38
bit 7, 23, 37
  data 30, 40, 93
  parity 40
  start 40
  stop 30, 40, 92, 94
block
  diagram, SSC 45
  moves 91
branch table locations, Pascal 1.1 50
break signal 36
buffer 45, 47
bus
  address 45
  data 45, 48
  parallel 48
byte 38
  device signature 51

C

cable
  connector 10, 26
  external 1, 10, 26
  internal 1, 9, 24, 45
  socket 2
capitalize 34
carriage return delay 7, 17, 94, 95, 96
carrier
  lost 110
  signal 42
chaining 34
character
  displays 33
  enquire 95
characters
  echo on screen 34
  lowercase 34, 102
  uppercase 34, 102
  capitalize 34
CIC 36, 91, 96
clamp assembly 2, 3
Clear To Send (see CTS)
code 38
column overflow 17
command
  character
    change 13, 28
    Communications Mode 28, 102

Printer Mode 12, 102
send 12
data format 15
format 12, 27
Communications Mode 27-28
register 45
remote-control 102
reset 15, 95
set time delay 16
summary, Terminal Mode 35
translate 91
commands
CIC 96
Communications Mode 27, 29, 102
Parallel Card 96
Printer Mode 11, 14
Terminal Mode 22, 35
communications 37
mode 21-33, 91, 95, 96
Communications Card, Apple II (see CIC)
Communications Mode 22, 26
command
character 28
summary 28-29
commands 27
control character 27
components SSC 45
configurations, SSC 44
connector
pin assignments 98
slots 9, 25, 98
control
character
communications mode 27
suppress 18, 33
logic 45, 46
register 45
<CR> 7, 17, 94, 95, 96, 102, 107
<CTRL-A> 27, 102
<CTRL-I> 8, 12, 102
<CTRL-R> 28, 35, 96, 102
<CTRL-S> 28, 96, 102
<CTRL-SHIFT-M> 101, 103
<CTRL-SHIFT-N> 103
<CTRL-T> 28, 35, 96, 102
CTS 41, 42, 43, 47, 48, 95, 98, 99

D

data 37
ASCII 24
binary 24
bit 7, 15, 23, 24, 30, 40, 93
bus 45, 48
bus buffer 45
conversion 39, 47
format 7, 23, 28, 30, 40, 47, 91, 92, 93
command 15
RS-232-C 40
selection 15, 23, 30
input 48, 96
output 48
serial 48
terminal 40
transfer 47
Data Carrier Detect (see DCD)
Data Communication Equipment (see DCE)
Data Set Ready (see DSR)
Data Terminal Equipment (see DTE)
Data Terminal Ready (see DTR)
DB-25 connector 10, 26, 97, 99
pin assignments 98
DCD 41, 42, 43, 47, 48, 95, 98, 99
DCE 41, 48, 105
decimal error codes 110
default parity 16
demodulator 41
device
class digit 51
identification 51
remote 19, 96
signature byte 51
DEVICE SELECT 46, 47
disable 12, 28
DOS 8, 11, 27
Dow Jones News & Quotes Reporter 22, 36
setup program for 36
DSR 41, 42, 43, 48, 95, 98, 99
DTE 41, 42, 48, 105
DTR 41. 42, 43, 95, 98, 99

E

E command 31, 34, 35
echoing 34, 35
edge connector 2
EIA 40
80-column-display 34
Electronic Industries
  Association (see EIA)
emulation
  modes 91-96
  old Serial Interface Card 91
  P8 92
  SIC 92, 102
enable 12, 28
enquire character 95
enter Terminal Mode 35
entry points
  BASIC 55
  hardware 49
  I/O routine 50
  monitor ROM 55
  Pascal 1.0 55, 56
  Pascal 1.1 55, 56
  SSC 55
environment, SSC 97
error
  codes 109-110
    BASIC 110
    Pascal 110
  framing 109, 110
  parity 109, 110
ENQ 95, 102
ENQ/ACK 102
<ESC> 34, 35, 101-104
<ESC>L 91
<ESC>U 91
ETX 91, 94, 95, 102, 106
ETX/ACK 91, 94, 102, 106
Expansion ROM, SSC 46, 52
even parity 16, 31, 40, 92, 93
external cable 1, 10, 26

F

F command 17, 31
FCC rules 10, 26
<FF> 14, 16, 17, 29, 32
firmware 34, 47, 49, 109
  entry points, Pascal 1.0
    (SIC) 49
  interrupt handling 50
  listings 57-96
  locations, special 57
  memory usage 53
  protocol, Pascal 1.1 49
  SSC driver 46
form feed delay 14, 16, 17,
  29, 32
format, data 92, 93
framing error 109, 110
full-duplex 35

G

generate
  ASCII characters 35
  <CR> 17, 95, 96
  <LF> 8, 17, 24, 31, 92, 94
generic signature 51
ground 10, 26, 42, 43

H

half-duplex 35, 36, 95
handshake 94, 95
handshaking 19
hardware 96
  entry points 49
  handshaking 19
hexadecimal 101

I

IN# 14
IDS 560 Paper Tiger 6
indirect addressing 56
input 37
  data 48, 96
installation, SSC 9
Integer BASIC 36
interface
  receive 45
  transmit 45
internal
  cable 1, 9, 24, 45
    attachment 3
    clamp assembly 2, 3
  shielding 10, 26
interrupt
  handling 50
  request 47
    Pascal 1.0 50

I/O
  routine entry points 50
  scratchpad address 109
  SELECT 46
  space
    peripheral card 52
    registers in 54
  STROBE 46
IORESULT 109

J

jumper block 2, 5, 11, 21, 27, 42, 45, 48, 95, 99, 105
  wiring 99

K

keyboard 11, 18, 27, 33, 101, 102

L

<LF> 8, 17, 24, 31, 91, 92, 94, 102
line width 8, 92, 94, 96
linefeed (see <LF>)
listings, firmware 57-96
local 37
lowercase 14, 18, 29, 32-33, 34, 91, 92, 93, 95, 101, 102
  character displays 18

M

MARK parity 16, 31, 40, 101
mask <LF> in 17, 31
memory usage map, SSC 53
mode
  communications 21-33, 26, 91, 96, 102
  emulation 91-96
  P8 93
  P8A 93-94, 102, 106
  printer 5-19, 91, 92, 93-94, 102
  SSC operating 43
  terminal 22, 34-36, 96, 101, 102
modem 22, 26, 27, 41, 42, 105
  answer 42
  eliminator 5, 42, 43, 45, 48, 99, 105
  null 42, 48
  originate 42
MODEM 5, 21, 27, 48, 99
modulator 41
monitor
  Apple II 11
  ROM entry points 55

N

NEC 5510 Spinwriter 6
no parity 31
null modem 42, 48

O

odd parity 16, 31, 40
offsets, Pascal 1.1 57
originate 40
  modem 42
oscillator 45, 47
output 37
  data 48
override switch settings 27
overrun 109, 110

P

P8 91-95
  emulation POKEs 92
  ROM 92, 94, 95
P8A 91
  mode 6, 102, 106
  ROM 92, 94, 95
parallel
  bus 48
  card commands 96
  data 37, 38-39
  I/O card 52
parity 7, 15, 23, 30, 47, 92, 93
  bit 7, 23, 24, 40
  error 109, 110
  even 40
  MARK 40, 101
  odd 40
  selection 16, 31

SPACE 40
Pascal 11, 12, 27
error codes 109, 110
Pascal 1.0 49
entry points 55, 56
interrupt requests 50
Pascal 1.1 49
branch table locations 50
device
class digit 51
identification 51
signature byte 51
entry points 55, 56
firmware protocol 49
I/O routines 50
offsets 50
offsets 57
PEEK 91, 92, 93-94, 109
peripheral I/O space 49
physical characteristics, SSC 97
pin assignments 98
POKE 19, 91, 92, 93-94, 95
power supply 9, 25
PR# 14
preparing SSC, printer mode 5
printer(s) 6
local 11
using 11
Printer Mode 5-19, 91, 92, 93-94, 95
command character 12, 102
commands 11
summary 13-14
data format 7, 15
switch settings 6
using 11
protocol 102
BASIC 50
Pascal 1.1 firmware 49

Q

Q command 35, 36
Qume 6
quit Terminal Mode 36, 102

R

RAM 45, 49, 92
scratchpad locations 53
receive
interface 45
register 42
Received Data (see RXD)
register
command 45, 54
control 45, 54
receive 42
status/reset 45, 54
transmit 42, 54
transmit/receive 45
registers
A 50, 55-57
ACIA 49, 54
Pascal 1.1 50
peripheral I/O space 47
switch settings 45
X 50, 55-57
Y 50, 55-57
remote 37
device 19, 22, 26, 27, 96
terminal 34
remote-control commands 102
Request To Send (see RTS)
reset 27
command 15, 95
SSC 17, 31
RESET 46
<RETURN> 12, 27, 102
Ring Indicator (RI) 41
ROM 45
SSC Expansion 46, 52
ROM/RAM
address mapping 46
space 46
RS-232-C 40, 43, 47, 48, 95, 105
signals 40-43, 49, 98-99
RTS 41, 42, 43, 95, 98, 99
RXD 41, 42, 43, 98, 99

S

schematic, SSC 45, 100
scratchpad RAM locations 53
SCTS 98, 99
Secondary Clear To Send (see SCTS)
selection
baud rate 15, 30
data format 15, 23, 30
parity 16, 31
time delay 16, 32

serial
  data 37, 39, 40, 48
  interface card 95
setting switches 6, 22, 92
settings, switch 6, 22, 93-94
shielding, internal 10, 26
SIC 91, 92
  emulation mode 92, 102
  firmware entry points 49
  P8A mode 102, 106
  switch settings 93, 94
6502 registers 49, 50, 55-57
slew rate 48
software compatability 98
SPACE parity 16, 31, 40, 101
special
  characters, ASCII 101
  circuits, SSC 97
specifications, SSC 97-98
specify screen slot 34
SSC 1, 91, 95, 96, 102
  block diagram 45
  cable connector 10
    preparation 21
  configurations 44
  connect
    printer 7
    terminal 8
  controlling 11
  driver firmware 46
  emulation 94
  entry points 50, 55-57
  environment 97
  Expansion ROM 46, 52
  firmware memory usage 53
  installation 9, 92
  lowercase 18
  main components 45
  operating modes 43
  peripheral card I/O space 52
  photo 1
  physical characteristics 97
  registers in I/O space 54
  reset 17, 31
  RS-232-C signals 99
  schematic 45, 100
  scratchpad RAM locations 53
  slot location 98
  special circuits 97
  specifications 97-98
  switch settings 93-94
  theory of operation 45
  unpacking 1
  using
    Communications Mode 27
    printer 11
    Printer Mode 11
    terminal 11
  zero-page locations 52
start bit 7, 23, 40
status
  byte (see STSBYTE)
  register, ACIA 109
status/reset register 45
stop bit 7, 15, 23, 24, 30, 40, 92, 94
STSBYTE 109
summary
  Communications Mode commands 28-29
  Printer Mode commands 14
  Terminal Mode commands 35
Super Serial Card (see SSC)
suppress
  characters
    control 33
    keyboard 18, 33
  <LF> in 17, 31
switch settings
  baud rate 7, 23
  change 14, 30
  Communications Mode 22
  line width 8
  override 12, 27
  Printer Mode 6
  SIC 93-94
  SSC 93-94
  video 8
switches 2
  Communications Mode 21-22
  setting 22, 92
  SSC 45
  state of 47
  SW1 48, 54, 56, 57, 92, 93
    SW1-1 through SW1-4 2, 7, 15, 23, 31, 92, 93, 106
    SW1-5 and SW1-6 2, 5, 21, 92, 93, 102, 115
    SW1-7 2, 8, 24, 48, 92, 99
  SW2 6, 22, 48, 54, 56, 57, 92, 93, 94
    SW2-1 2, 7, 15, 23-24, 31, 92, 93, 94, 106
    SW2-2 2, 7, 16, 23-24, 31, 92, 94, 106
    SW2-3 and SW2-4 2, 8, 23, 92, 94, 106, 107

SW2-5 2, 8, 17, 24, 32, 92, 93, 106
SW2-6 2, 8, 24, 48, 92, 107
SW2-7 2, 8, 24, 48, 92, 99

T

T command 35
tab 19
terminal
connecting 8
local 11
unintelligent 34, 35
TERMINAL 5, 11, 21, 42, 48, 95, 99, 101, 105
Terminal Mode 22, 34-36, 101-104
bypass 96
change to 96
commands 35
enter 35, 102
example 36
quit 36, 102
theory of operation, SSC 45
time delay
selection 16, 32
time-sharing 34, 36
translate 94
command 17, 31, 91
lowercase 18
transmit
break signal 36
interface 45
register 42
transmit/receive register 45
Transmitted Data (see TXD)
troubleshooting 105-107
at Apple II 107
at device 106
TXD 41, 42, 43, 98, 99

U

unintelligent terminal 34, 35, 36
uppercase 14, 18, 29, 33, 34, 91, 92, 102

V

video on/off 8, 92, 94, 96

W

wiring, jumper block 99

X

X command 17, 31, 35, 106
X register 50, 55-57
XOFF 35, 102
recognition 18, 33
XON 102
XON/XOFF 17, 19, 31, 33, 96, 102

Y

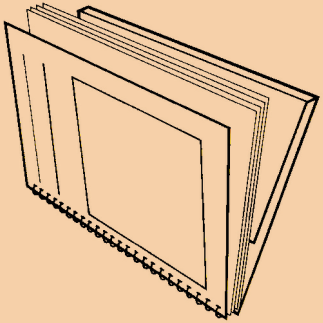
Y register 50, 55-57

Z

Z command 17, 18, 31
zero-page locations, SSC 52

Downloaded from www.Apple2Online.com

Tuck end flap
inside back cover
when using manual.



10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

030-0270-A